AWS SAA

Section I
- IAM & CLI
- EC2
- EC2 Instance Storage
- High Availability & Scalability
- RDS, Aurora & ElastiCache
- Route 53

Section II
- Amazon S3
- Amazon S3 Security
- CloudFront, Global Accelerator & Wavelength
- AWS Storage Extras
- Decoupling Applications
- Containers on AWS
- AWS Serverless

Section III
- Databases in AWS
- Data & Analytics
- Machine Learning
- AWS Monitoring & Audit
- IAM - Advanced

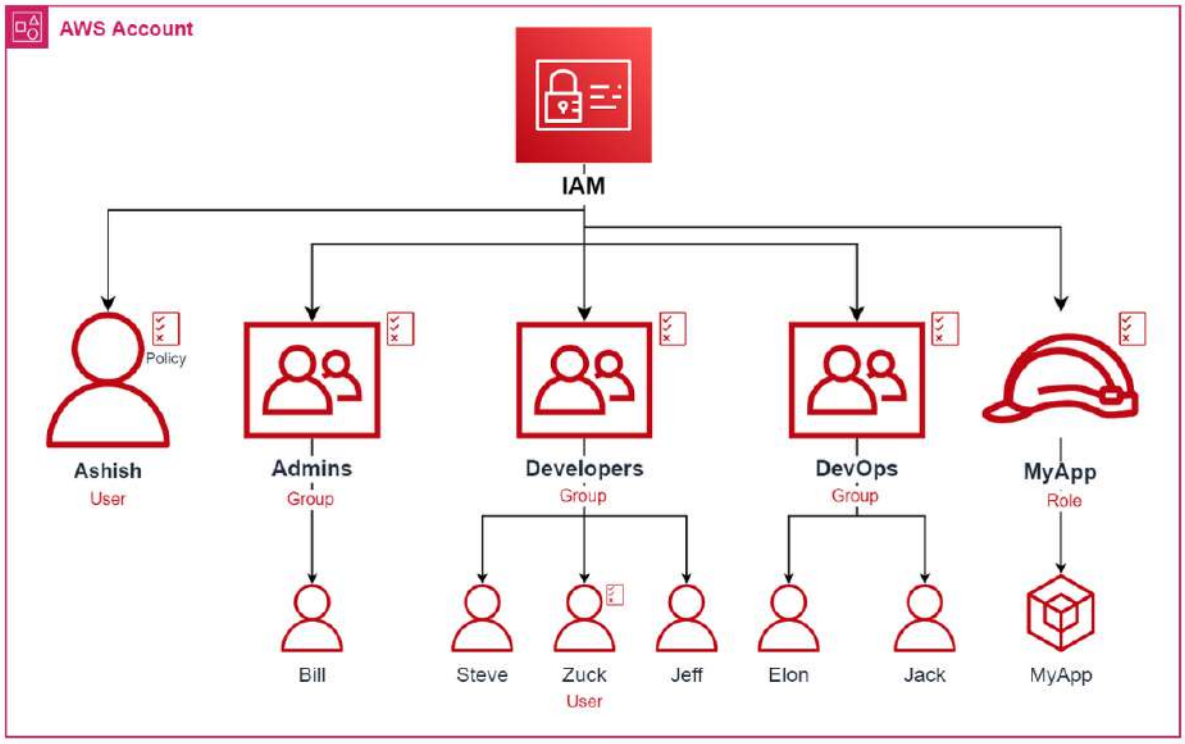
Section IV

AWS Security & Encryption
Networking - VPC
Disaster Recovery & Migrations
Other Services
WhitePapers & Architectures
Other

Section I

IAM & CLI

▼ ☑ **IAM Users, Groups and Policies.**



Root Account - created by default, shouldn't be used or shared.

Users - people within our organization and they can be grouped. Users dont have to belong to a group and one user can belong to multiple groups. Each user has a unique name and credentials (password or access keys).

Groups - collections of users. Instead of attaching policies to individual users, we can assign permissions to groups, making it easier to manage access based

on job function or role.

Users or Groups can be assigned JSON documents called policies which define the permissions of the user. In AWS we apply the **least privilege principle** (dont give more permissions than a user needs).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:Describe*"
            ],
            "Resource": "*"
        }
    ]
}
```

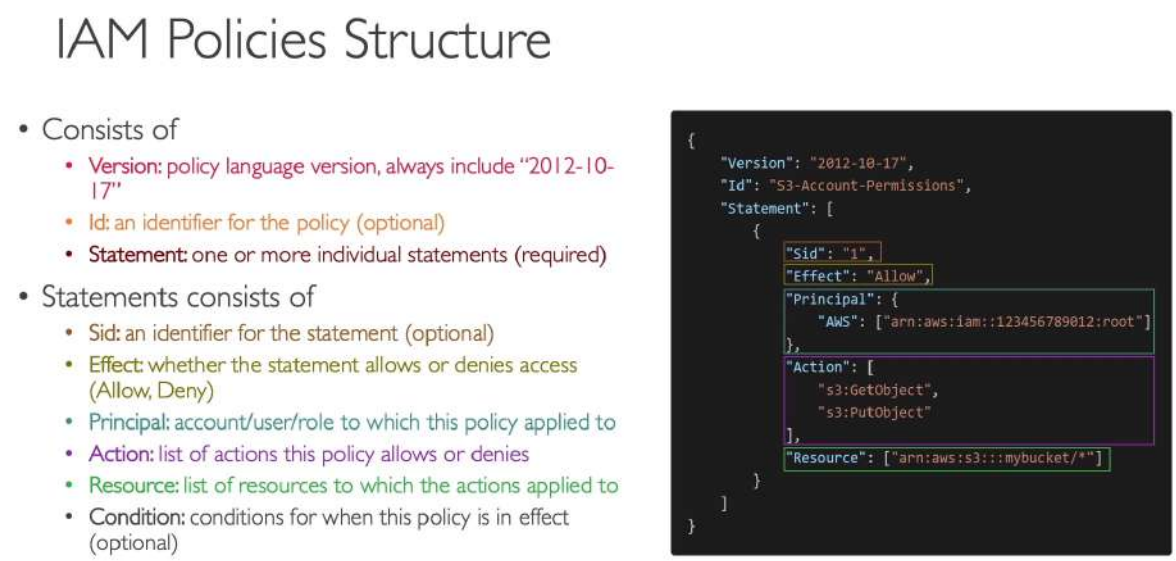
There are also **Roles** which give permissions to AWS services.

▼ ☑ **IAM Policies Structure.**

IAM Policy Components

- Statement - main element of an IAM policy, and it consists of one or more individual statement.
- Sid - identifier for the statement.
- Effect - specifies whether the statements allows or denies the specified action.
- Principal (Optional) - Specifies the entity (user, account or role) that the policy is applied to.
- Action - lists the specific actions that the policy allows or denies. Actions are represented as strings, often in the format "**[service]:[action]**".

- Resource - specifies the AWS resources.
- Condition (Optional) - defines conditions under which the policy statement is in effect.



▼ ☑ IAM MFA.

In AWS we can set minimum password length, specific character types as requirement, allow IAM users to change their passwords after some time and prevent password re-use.

MFA (Multi Factor Authentication) = password + security device.

- Virtual MFA (Google Authenticator, Authy)

  Support for multiple tokens on a single device.

- U2F (Universal 2nd Factor) (YubiKey)

  Support for multiple root and IAM users using a single security key.

- Hardware Key Fob MFA Device (Gemalto)
- Hardware Key Fob MFA Device for AWS GovCloud (SurePassID)

▼ ☑ AWS Access Keys, CLI & SDK, CloudShell.

Access Keys are credentials used to authenticate and authorize API requests to AWS services. These keys consist of:

1. **Access Key ID:** Unique identifier that AWS uses to identify the IAM user or AWS account making the API request.

2. **Secret Access Key**: Secret key known only to the IAM user or AWS account. It is used to digitally sign API requests made by the user, ensuring that the request is authentic and comes from a trusted source.

Three options to access AWS:

- **AWS Management Console** (Protected by MFA)

  Web-based interface provided by AWS that allows users to access and manage their AWS services and resources.

- **AWS CLI** (Protected by access keys)

  A tool that enables interaction with AWS services using commands in command line shell. It lets us to direct access public APIs of AWS services. We can use it to develop script to manage our resources.

- **AWS SDK** (Protected by access keys)

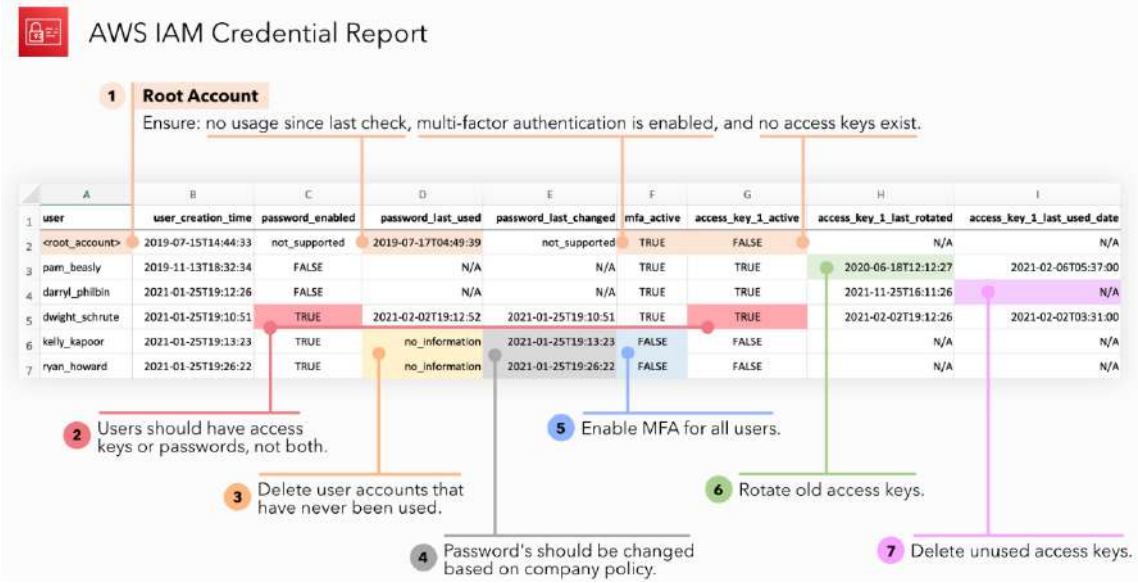
  It consists of libraries that let us to access and manage AWS services progammatically.

Access keys are generated through the AWS Console. Users manage their own access keys.

AWS CloudShell is a browser-based shell environment that allows us to manage and interact with AWS resources directly from web browser. It's not available in all regions.

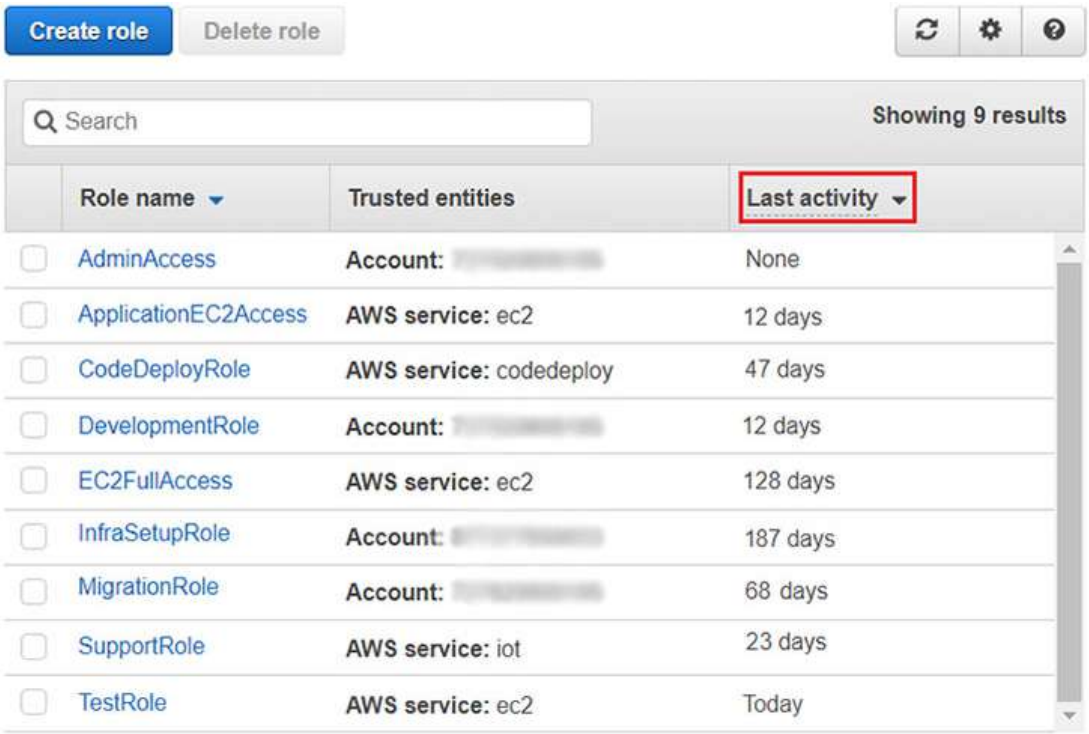
▼ ☑ **IAM Security Tools.**

- **IAM Credential Report** (**account-level**) - Report that lists all account's users and the status of their credentials.



user	user_creation_time	password_enabled	password_last_used	password_last_changed	mfa_active	access_key_1_active	access_key_1_last_rotated	access_key_1_last_used_date
<root_account>	2019-07-15T14:44:33	not_supported	2019-07-17T04:49:39	not_supported	TRUE	FALSE	N/A	N/A
pam_beasly	2019-11-13T18:32:34	FALSE	N/A	N/A	TRUE	TRUE	2020-06-18T12:12:27	2021-02-06T05:37:00
darryl_philbin	2021-01-25T19:12:26	FALSE	N/A	N/A	TRUE	TRUE	2021-11-25T16:11:26	N/A
dwight_schrute	2021-01-25T19:10:51	TRUE	2021-02-02T19:12:52	2021-01-25T19:10:51	TRUE	TRUE	2021-02-02T19:12:26	2021-02-02T03:31:00
kelly_kapoor	2021-01-25T19:13:23	TRUE	no_information	2021-01-25T19:13:23	FALSE	FALSE	N/A	N/A
ryan_howard	2021-01-25T19:26:22	TRUE	no_information	2021-01-25T19:26:22	FALSE	FALSE	N/A	N/A

- **IAM Access Advisor** (**user-level**) - Shows the service premissions granted to a user and when those services were last accessed. This information can be used to revise policies.



EC2

▼ ☑ **EC2 Basics.**

EC2 is an IaaS that offer:

- Renting virtual machines (EC2 Instances)
- Storing data on virtual drives (EBC)
- Distribute load across machines (ELB)
- Scale the services using an auto-scaling group (ASG)

When we create EC2 we can configure OS, CPU, RAM, storage, network card, security and bootstrap script.

Bootstrap script is a configuration script that runs at first launch. Bootstraping instances is done by using **EC2 User Data** script (**runs with root user privileges**).

Instance metadata is data about our EC2 instance that we can use to configure or manage the running instance. Because our instance metadata is available from our running instance, we do not need to use the EC2 console or the AWS CLI. This can be helpful when we're writing scripts to run from your instance.

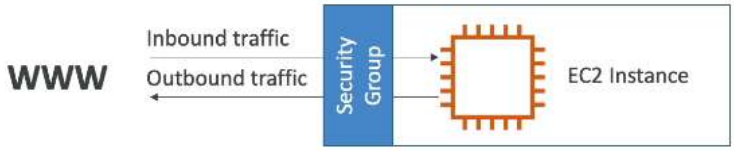
▼ ☑ **EC2 Instance Types.**

	Type	Description	Mnemonic
General Purpose	a1	Good for scale-out workloads, supported by Arm	**a** is for Arm processor – or as light as **A1** steak sauce
	t-family: t3, t3a, t2	Burstable, good for changing workloads	**t** is for **tiny** or **turbo**
	m-family: m6g, m5, m5a, m5n, m4	Balanced, good for consistent workloads	**m** is for **main** or happy **medium**
Compute Optimized	c-family: c5, c5n, c4	High ratio of compute to memory	**c** is for **compute**
Memory Optimized	r-family: r5, r5a, r5n, r4	Good for in-memory databases	**r** is for **RAM**
	x1-family: x1e, x1	Good for full in-memory applications	**x** is for **xtreme**
	High memory	Good for large in-memory databases	High memory is for... high memory.
	z1d	Both high compute and high memory	**z** is for **zippy**
Accelerated Computing	p-family: p3, p2	Good for graphics processing and other GPU uses	**p** is for **pictures**
	Inf1	Support machine learning inference applications	**Inf** is for **inference**
	g-family: g4, g3	Accelerate machine learning inference and graphics-intensive workloads	**g** is for **graphics**
	f1	Customizable hardware acceleration with field programmable gate arrays (FPGAs)	**f** is for **FPGA** or **feel** as in hardware
Storage Optimized	i-family: i3, i3en	SDD-backed, balance of compute and memory	**i** is for **IOPS**
	d2	Highest disk ratio	**d** is for **dense**
	h1	HDD-backed, balance of compute and memory	**H** is for **HDD**

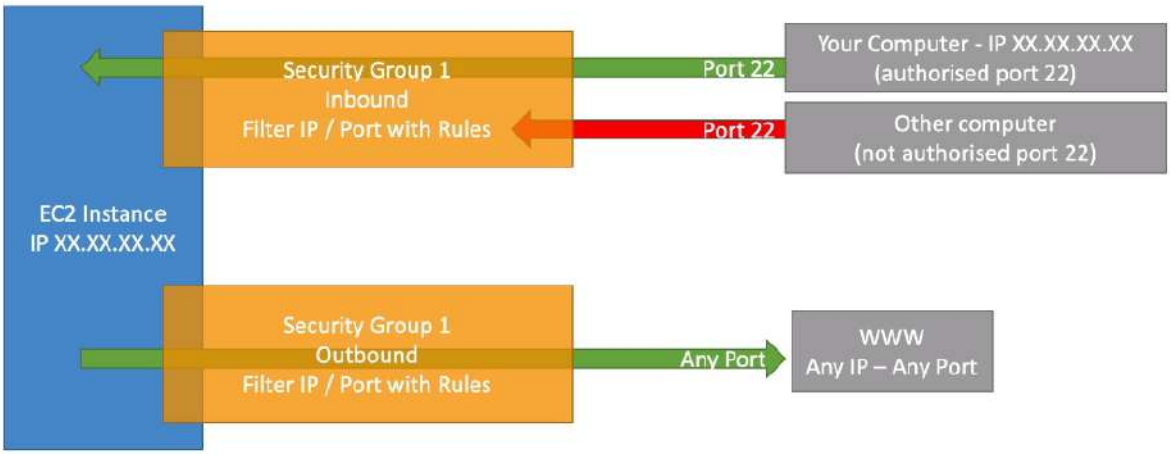
▼ ☑ Security Groups.

Security Groups are the fundamental of network security in AWS. They control how traffic is allowed INTO or OUT of our EC2 Instances.

Security groups only contain allow rules. Rules can reference by IP or by another security group.



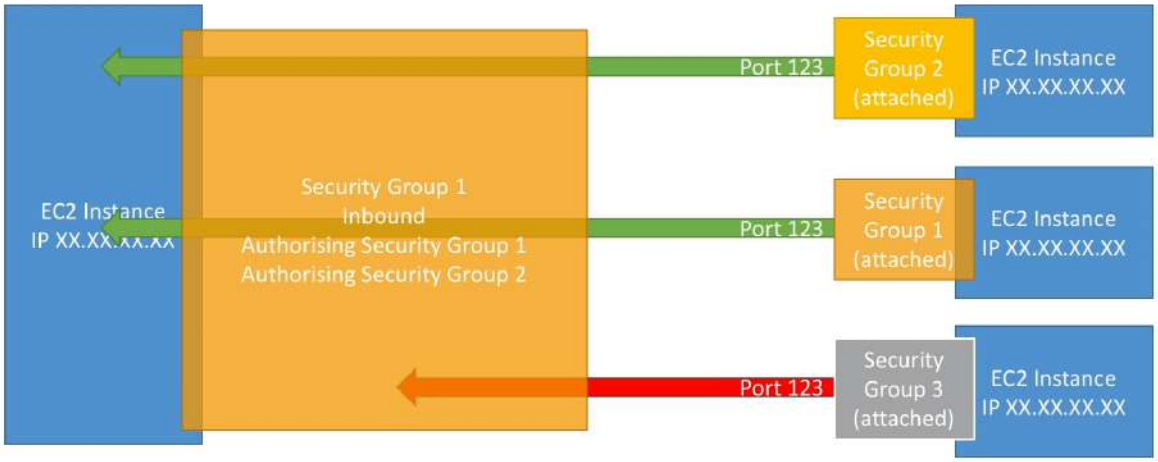
Security groups are acting as a "firewall" on EC2 instances. They regulate access to ports, authorised IP ranges, control of inbound network and control of outbound network.



Security groups can be attached to multiple instances. They are locked down to a region/VPC combination.

It's good practice to maintain one sparate security group for SSH access.

If application is not accessible (time out), then it is a security group issue and if application gives a "connection refused", then it is an application error.



▼ ☑ **EC2 Instance Connect.**

EC2 Instance Connect is a feature that allows us to connect to our EC2 instances without needing to manage SSH key. The user must have the necessary IAM permissions to use EC2 Instance Connect.

It reduces the risk associated with managing long-term SSH keys.

▼ ☑ **EC2 Launching Types.**

- **On Demand Instances**: short workload, predictable pricing.

  We pay for what we use. Has the highest cost but no upfront payment and no long-term commitment.

  Recommended for short-term and un-interrupted workloads, where we cant predict how the application will behave.

- **Reserved**: (1 & 3 years)
  - **Reserved Instances**: long workloads.

    Up to 72% discount compared to On-demand. We reserve a specific instance attributes (Instance Type, Region, Tenancy, OS). We can buy and sell in the Reserved Instance Marketplace.

    Reservation Period:

    - 1 year (Discount +)
    - 3 years (Discount +++)

    Payment Options:

        - No Upfront (Discount +)
        - Partial Upfront (Discount ++)
        - All Upfront (Discount +++)

      Reserved Instance's scope can be regional or zonal.

    - **Convertible Reserved Instances**: long workloads with flexible instances.

      Can change the EC2 instance type, instance family, OS, scope and tenancy. Up to 66% discount.

- **Savings Plans** (1 & 3 years): commitment to an amount of usage, long workload.

  Get a discount based on long-term usage (up to 72%). We commit to a certain type of usage ($10/hour for 1 or 3 years). Usage beyond Saving Plans is billed at the On-Demand price.

  Locked to a specific instance family & AWS Region, but flexible across instance size, OS and tenancy.

- **Spot Instances**: short workloads, for cheap, can lose instances.
- Can get a discount up to 90% compared to On-demand. These are instances that you can lose at any point of time if your max price is less than the current spot price. They are not suitable for critical jobs or databases.

  Useful for workloads that are resilient to failure such as: batch jobs, data analysis, image processing, any distributet workloads ...

- **Dedicated Instances**: no other customers will share our hardware.

  Instances run on hardware that's dedicated to us. May share hardware with other instances in same account.

- **Dedicated Hosts**: book an entire physical server, control instance placement.

  A physical server with EC2 instance capacity full dedicated to our use. Allows us to address compliance requirements and use existing server-bound software licenses (per-socket, per- core). This is the most expensive option.

  Userful for software that have complicated licensing model and for companies that have strong regulatory or compliance needs.

Purchasing Options:

- On-demand: pay per second for active dedicated host.
- Reserved: 1 or 3 years

Characteristic	Dedicated Instances	Dedicated Hosts
Enables the use of dedicated physical servers	X	X
Per instance billing (subject to a $2 per region fee)	X	
Per host billing		X
Visibility of sockets, cores, host ID		X
Affinity between a host and instance		X
Targeted instance placement		X
Automatic instance placement	X	X
Add capacity using an allocation request		X

- **Capacity Reservations**: reserve capacity in a specific AZ for any duration.

  Reserve On-Demand instances capacity in a specific AZ for any duration. We always have access to EC2 capacity when we need it. No time commitment and no billing discounts. To benefit from billing discounts we can combine it with Regional Reserved Instances and Saving Plans.

  We are charged at On-Demand rate whether we run instances or not.

  Suitable for short-term, uninterrupted workloads that needs to be in a specific AZ.

▼ ☑ **EC2 Placement Groups.**

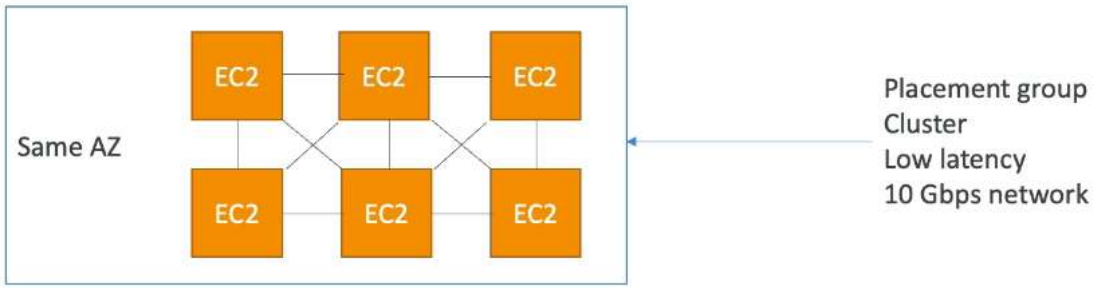
EC2 Placement Groups are a feature that enables us to influence the placement of Amazon EC2 instances on the underlying hardware. This can be important for workloads that require high performance, low latency, or other specific characteristics.

If we receive a capacity error when launching an instance in a placement group that already has running instances, stop and start all of the instances in the placement group, and try the launch again. Restarting the instances may migrate them to hardware that has capacity for all the instances.

- **Cluster Placement Group**

  Designed for applications that need low network latency, high network throughput, or both and for Big Data jobs that need to complete fast.

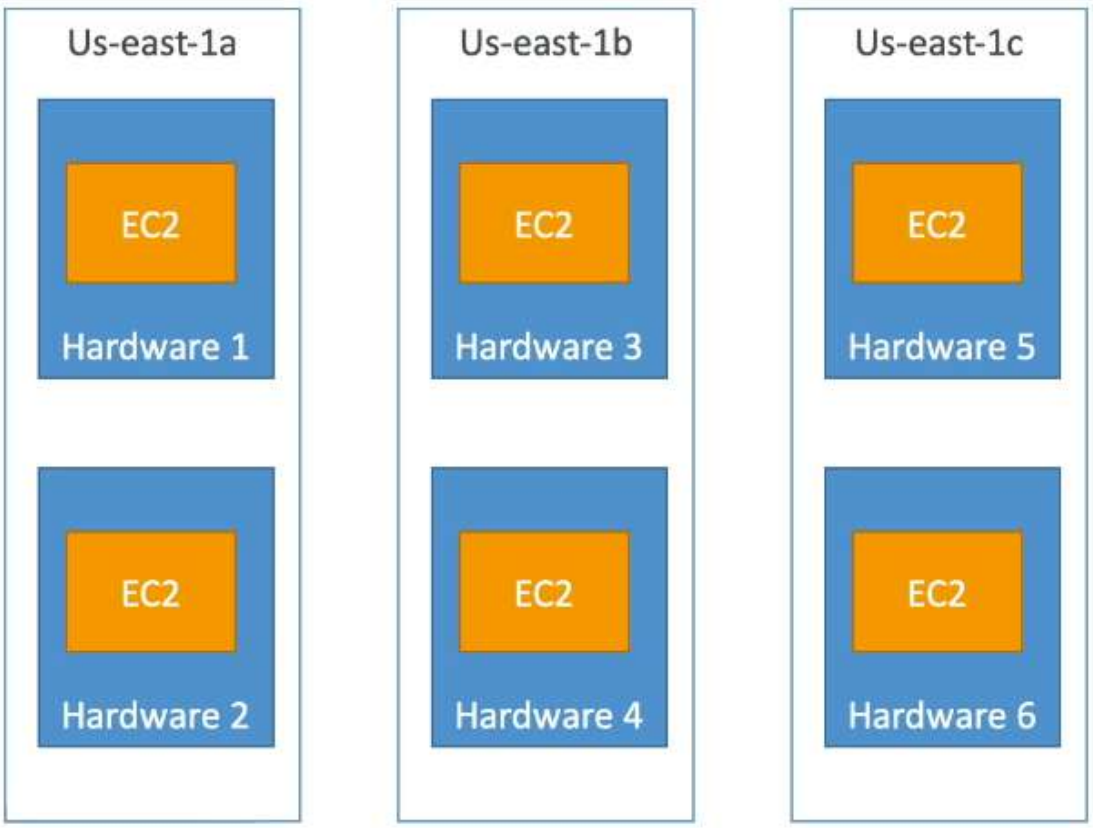
  All instances must be in the same AZ and there might be limits on the number of instances we can place in a single group depending on the instance type and available hardware.



- **Spread Placement Group**

  Suited for applications that require high availability.

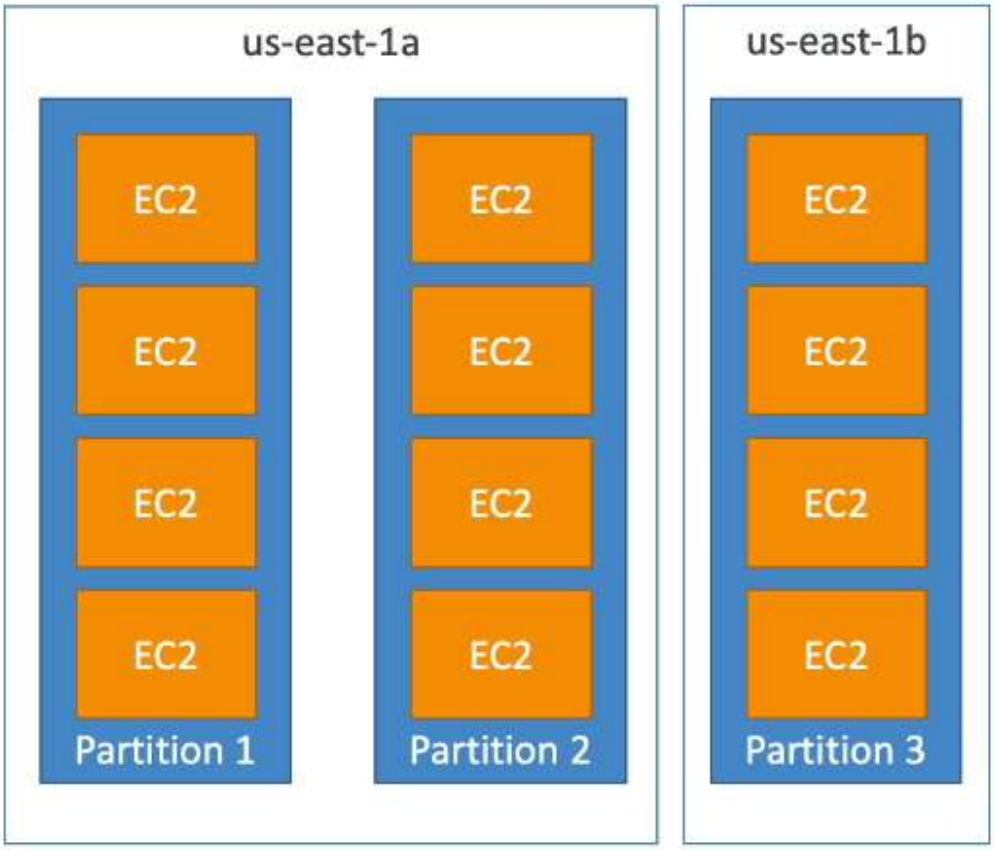
  Instances are distributed across distinct underlying hardware. This minimizes the risk of correlated hardware failures. Spread placement groups have stricter limits on the number of instances per AZ, typically up to seven.



- **Partition Placement Group**

  Useful for large-scale distributed and replicated workloads, such as Hadoop, Cassandra, and Kafka.

  Instances are divided into logical partitions, each isolated from the other in terms of failure. Failures in one partition do not affect the instances in other partitions. We have more control over where instances are placed, but the number of partitions and instances per partition can be constrained.



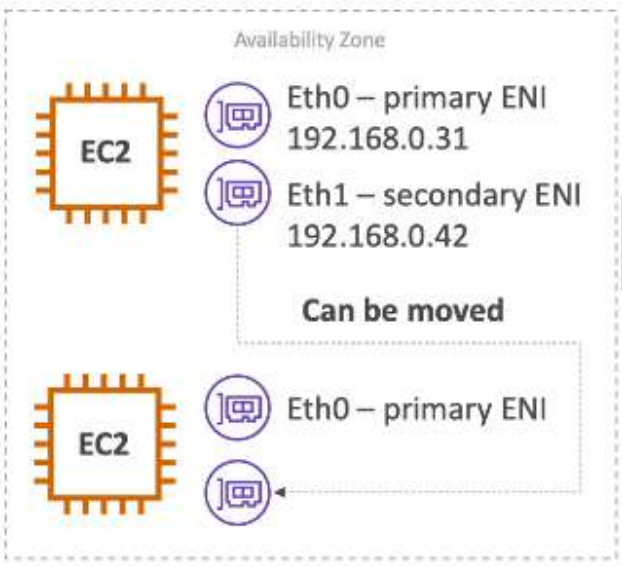
▼ ☑ **Elastic Network Interface (ENI).**

ENI is a logical networking component in a VPC that represents a virtual network card.

ENI Attributes:

- **Primary Private IP Address**
- **Secondary Private IP Addresses**
- **One Elastic IP Addresses**
- **One Public IPv4 Address**
- **MAC Address**
- **Security Groups**

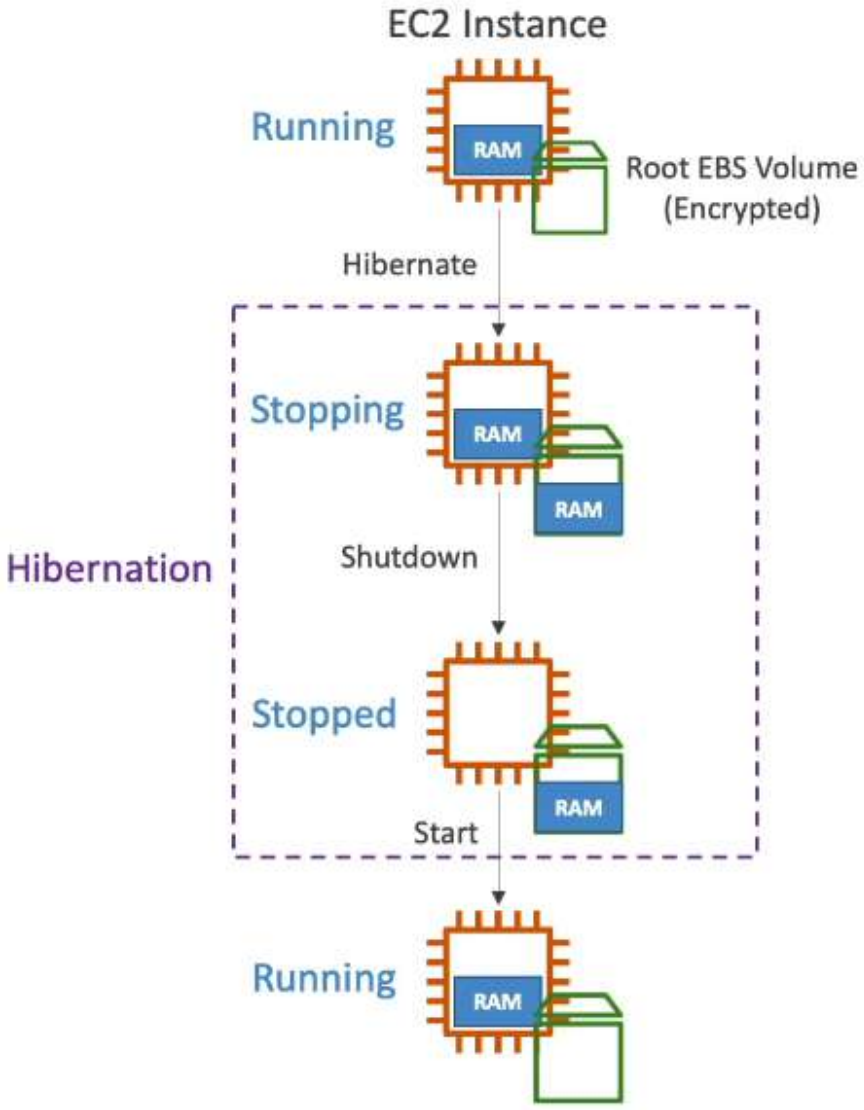
ENIs can be attached to instances in the same VPC, but must reside in the same AZ. We can attach a secondary ENI to an instance and use it for **failover** scenarios. If the primary instance fails, we can attach the ENI to a standby instance.



▼ ☑ **EC2 Hibernate.**

Hibernate is a feature designed to help us manage our EC2 instances more efficiently by allowing us to pause and resume them. When we hibernate an instance, the contents of its memory (RAM) are saved to an EBS volume, and the instance is stopped rather than terminated. We can later resume the instance from this saved state. The root EBS volume must be encrypted.

Hibernation is useful for workloads that don't require continuous operation but need to be quickly resumed.



EC2 Instance Storage

▼ ☑ **AMI.**

AMI are customization of an EC2 instance. We can add our own configuration, OS, software etc... All our software is pre-packaged so we have faster boot.

AMI are built for a specific region and can be copied across regions.

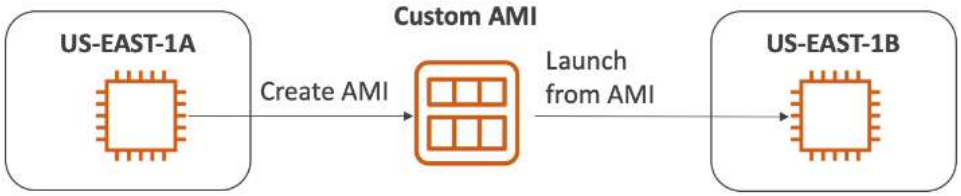
EC2 instances can be launched from:

- Public AMI - AWS provided.
- Our own AMI - we make and maintain them.

- AWS Marketplace AMI - an AMI someone else made.

AMI Process (from an EC2 instance)

- Start an EC2 instance and customize it
- Stop the instance (for data integrity)
- Build an AMI – this will also create EBS snapshots
- Launch instances from other AMIs



▼ ☑ **EBS.**

EBS Volume (think of them as a "network USB stick") is a network drive we can attach to our instances (cloud only) while they run. It allows our instances to persist data even after their termination. EBS volumes support live configuration changes while in production which means that we can modify the volume type, volume size, and IOPS capacity without service interruptions.

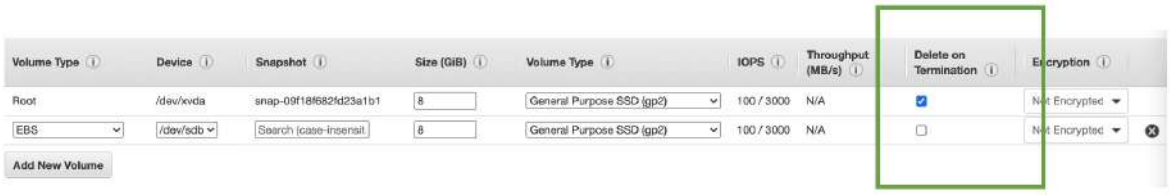
EBS can only be mounted to one instance at a time. They are bound to a specific AZ.

- It's a network drive (i.e. not a physical drive)
  - It uses the network to communicate the instance, which means there might be a bit of latency
  - It can be detached from an EC2 instance and attached to another one quickly
- It's locked to an Availability Zone (AZ)
  - An EBS Volume in us-east-1a cannot be attached to us-east-1b
  - To move a volume across, you first need to snapshot it
- Have a provisioned capacity (size in GBs, and IOPS)
  - You get billed for all the provisioned capacity
  - You can increase the capacity of the drive over time

Data Lifecycle Manager (DLM) - automate the creation, retention, and deletion of snapshots taken to back up our EBS volumes.

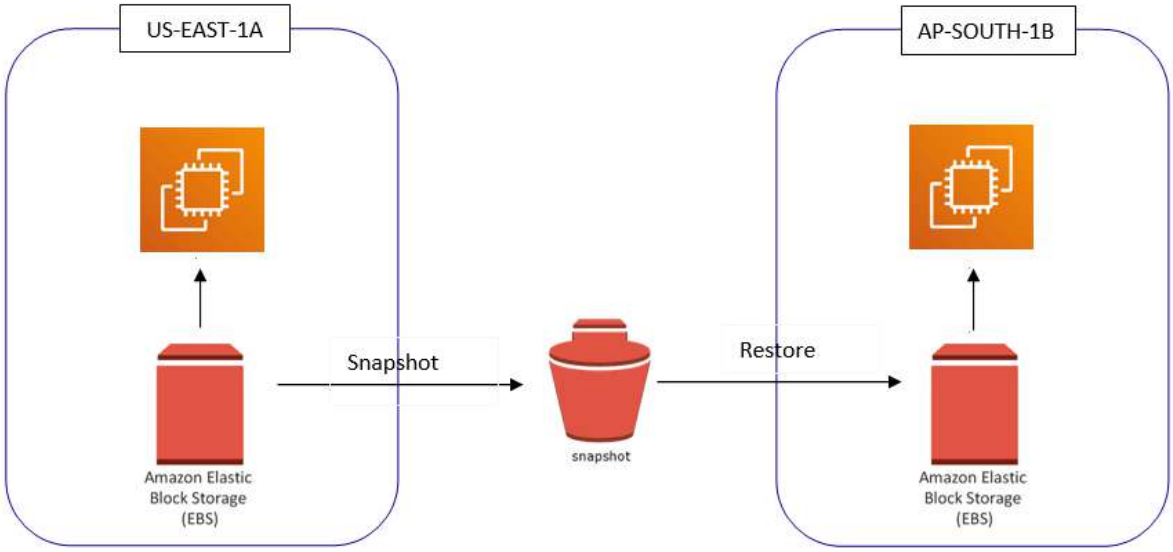
Delete on Termination

- Controls the EBS behaviour when an EC2 instance terminates.
  - By default, the root EBS volume is deleted (attribute enabled).
  - By default, any other attached EBS volume is not deleted (attribute disabled).
- This can be controlled by the AWS console / AWS CLI.
- Use case: preserve root volume when instance is terminated.



▼ ☑ **EBS Snapshots.**

It allows us to make a backup of EBS volume at a point in time. We can copy snapshots across AZ or Region. The EBS volume can be used while the snapshot is in progress.



EBS Snapshot Features:

- **EBS Snapshot Archive**

  We can move a snapshot to an “archive tier” that is 75% cheaper. It takes within 24 to 72 hours for restoring the archive.

- **Recycle Bin for EBS Snapshots**

  Setup rules to retain deleted snapshots so we can recover them after an accidental deletion.

▼ ☑ **EC2 Instance Store.**

EBS volumes have “limited” performance. If we need high performance hardware disk, we should use **EC2 Instance Store**.

EC2 Instance Store has better I/O performance and they are good for buffer, **cache**, scratch data or temporary content.

EC2 Instance Store lose their storage if they are stopped (ephemeral). So we have risk of data loss if hardware fails. Backups and replication are our responsibility.

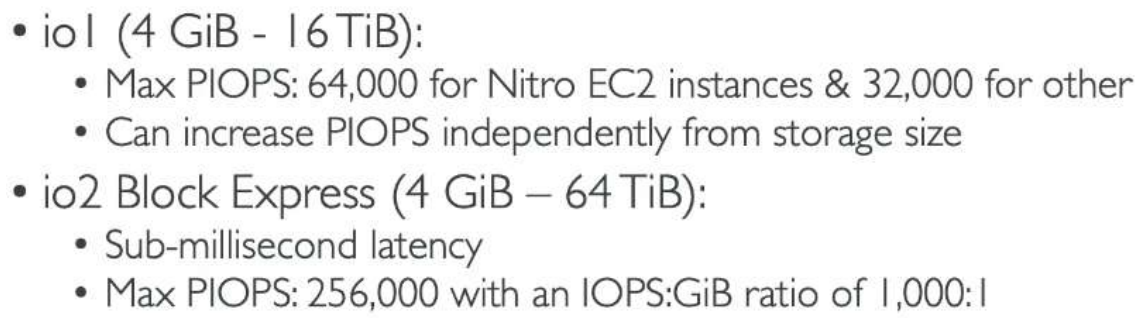
▼ ☑ **EBS Volume Types.**

EBS Volumes are characterized in size, throughput and IOPS. Only gp2/gp3 and io1/io2 Block Express can be used as boot volumes.

- **gp2 / gp3 (SSD)** - general purpose SSD volume that balances price and performance for a wide variety of workloads. Small gp2 volume can burst

IOPS to 3000. Gp3 is cheaper.

- **io1 / io2 Block Express (SSD)** - highest performance SSD volume for mission-critical low-latency or high-throughput workloads. Supports multi-attach. It's great for database workloads.



- **st1 (HDD)** - low cost HDD volume designed for frequently accessed, throughput-intensive workloads.
- **sc1 (HDD)** - lowest cost HDD volume designed for less frequently accessed workloads.

Volume Type	General Purpose SSD	Provisioned IOPS SSD	Throughput Optimized HDD	Cold HDD	EBS Magnetic HDD
Description	Balance price and performance for a wide variety of transactional workloads	Designed for mission critical applications which requires high performance	Low cost. Designed for frequently accessed throughput intensive workloads	The lowest cost. Designed for less frequently accessed workloads	Previous generation HDD
API Name	gp2, gp3	Io1, io2	St1	Sc1	Standard
Use Case	Most workloads	Large Database workloads	Big Data, Warehouses, Log processing	File Servers	Workloads with infrequently accessed data
Size	1GB - 16TB	4GB – 16TB	500GB - 16TB	500GB - 16TB	1GB - 1TB
Max IOPS	16,000	64,000	500	250	40-200
Can be a boot volume?	Yes	Yes	No	No	No
Bursting	gp2 = Yes gp3 = No	No	Yes	Yes	No

▼ ☑ **EBS Multi-Attach.**

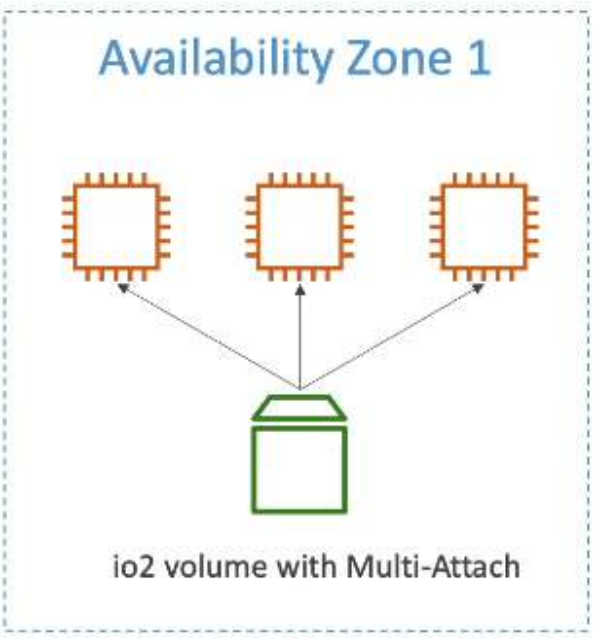
Multi-Attach lets us to attach same EBS volume to multiple EC2 instances in the same AZ. It is only possible with io1/io2 volume types. Each EC2 instance has full

read and write premissions to the high-performance volume.

It's **limited to 16 EC2 instnaced at a time** and we must use a file system that's cluster-aware (not XFS, EXT4, etc).

Use cases:

- Achieving higher application availability in clustered Linux applications.
- If applications must manage concurent write operations.



▼ ☑ **EBS Encryption.**

When we create an encrypted EBS volume:

- Data at rest is encrypted inside the volume.
- All the data in flight moving between the instance and the volume is encrypted.
- All snapshots are encrypted.
- All volumes created from the snapshot are encrypted.

Encryption has a minimal imapct on latency. EBS encryption leverages keys from KMS (AES-256).

▼ ☑ **RAID Configuration.**

RAID (Redundant Array of Independent Disks) is a technology that combines multiple physical disk drives into a single logical unit to provide data redundancy, performance improvements, or both.

- **RAID 0 (Striping)** - splits data evenly across two or more disks without any redundancy. The main goal is to **increase performance** by allowing read and write operations to happen in parallel across multiple disks.

  It is ideal for applications where performance is critical, but data loss is not a concern, such as video editing or gaming.

- **RAID 1 (Mirroring)** - duplicates the same data on two or more disks. The goal here is to provide **fault tolerance**. If one disk fails, the system can still operate with the remaining disk(s) containing an exact copy of the data.

  It is suitable for systems where data reliability is critical, such as database storage, web servers, or small business file servers.

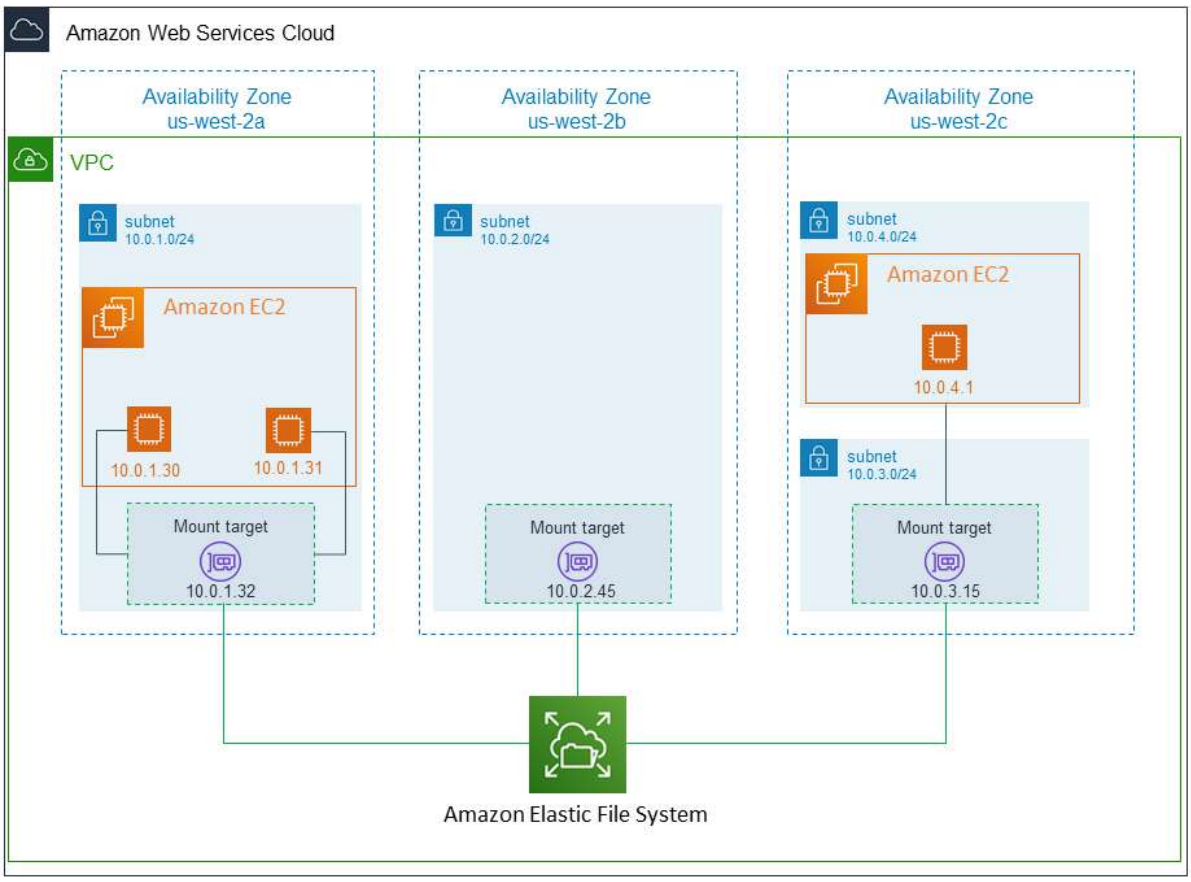
RAID Level	Performance	Redundancy	Capacity	Use Case
RAID 0	High	None	Full	Performance-critical, non-critical data
RAID 1	Moderate	Full	Half	Data reliability, critical systems

▼ ☑ **EFS.**

EFS is managed NFS (network file system) that can be mounted on hundreds of EC2. EFS works with **Linux** (uses POSIX file system) EC2 instances in multi-AZ. It uses NFSv4.1 protocol.

We can enable encryption at rest using KMS.

EFS is highly available, scalable, expensive, pay per use and has no capacity planning.



EFS can handle 1000s of concurrent NFS clients. Grows to petabyte-scale network file system automatically.

Performance Classes

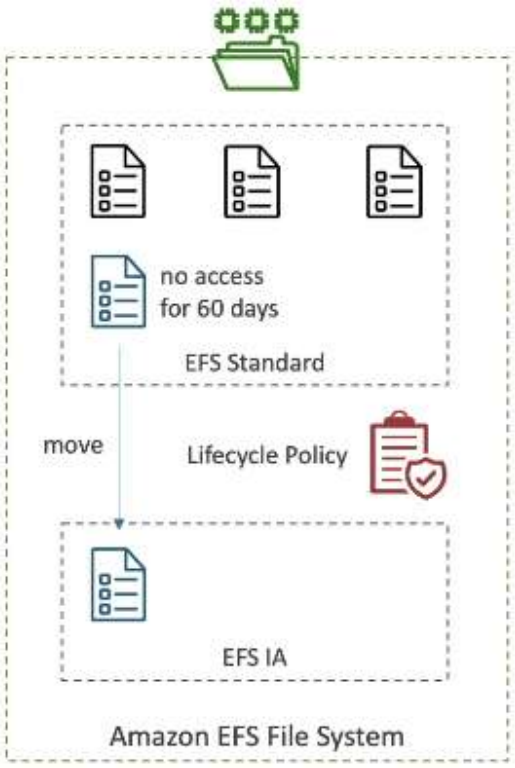
- **Performance Mode**
  - **General Purpose (default)** - best for latency-sensitive applications, such as web serving, content management systems, and development environments.
  - **Max I/O** - ideal for applications requiring high levels of aggregate throughput and parallelism, such as big data and media processing.
- **Throughput Mode**
  - **Bursting Throughput** - throughput scales with the size of the file system. Suitable for most file systems, offering a balance between cost and performance.

- **Provisioned Throughput** - allows us to specify the desired throughput level, which can exceed what is available via the burst mode. Best for applications requiring a consistent and predictable level of throughput.
- **Elastic Troughtput** - automatically scales up or down based on our workloads. Used for unpredictable workloads.

Storage Classes

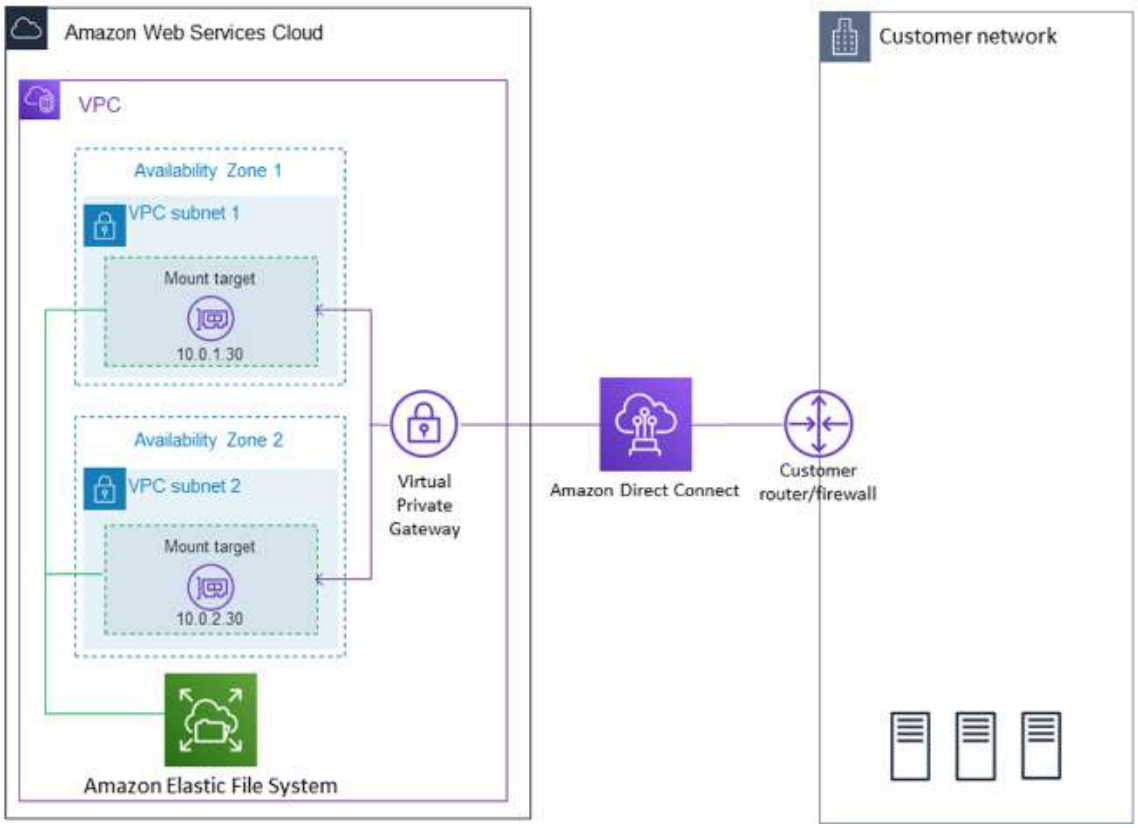
- **Standard** - for frequently accessed files. Data is redundantly stored across multiple Availability Zones within an AWS Region.
- **Infrequent Access (IA)** - for files that are accessed less frequently but require the same durability and availability as the EFS Standard class. It has lower price to store, but has retrieval fee.
- **Archive** - for rarelly accessed data (few times each year). It is 50% cheaper.

We need to implement lifecycle policies to move files between storage tiers.



▼ **Solution Architecture - Mounting EFS with on-premises Linux Clients.**

We can mount our EFS file systems on our on-premises data center servers when connected to our VPC with DX or VPN.



High Availability & Scalability

▼ ☑ **Scalability & High Availability.**

Scalability means that an application or system can handle greater loads by adapting. Scalability is linked but different to High Availability. There are two types of scalability:

- **Vertical Scalability**

  Increasing the size of the instance (**scale up/down**). Example: If we have application that runs on a t2.micro, scaling that application vertically means running it on t2.large.

  Vertical Scalability is very common for non distributed systems, such as a database.

- **Horizontal Scalability (Elasticity)**

Increasing the number of instances/systems for our application (**scale out/in**). Horizontal scaling implies distributed systems (very common for web applications).

High Availability means running our application or system in at least 2 AZ. High Availability usually goes hand in hand with horizonntal scalling. The main goal of high availability is to survive data center loss (disaster).

Scalability vs Elasticity

- Scalability - ability to accommodate a larger load by making the hardware stronger (scale up) or by adding nodes (scale out).
- Elasticity - once a system is scalable, elasticity means that there will be some “auto-scaling” so that the system can scale based on the load (**automatic horizontal scaling**).

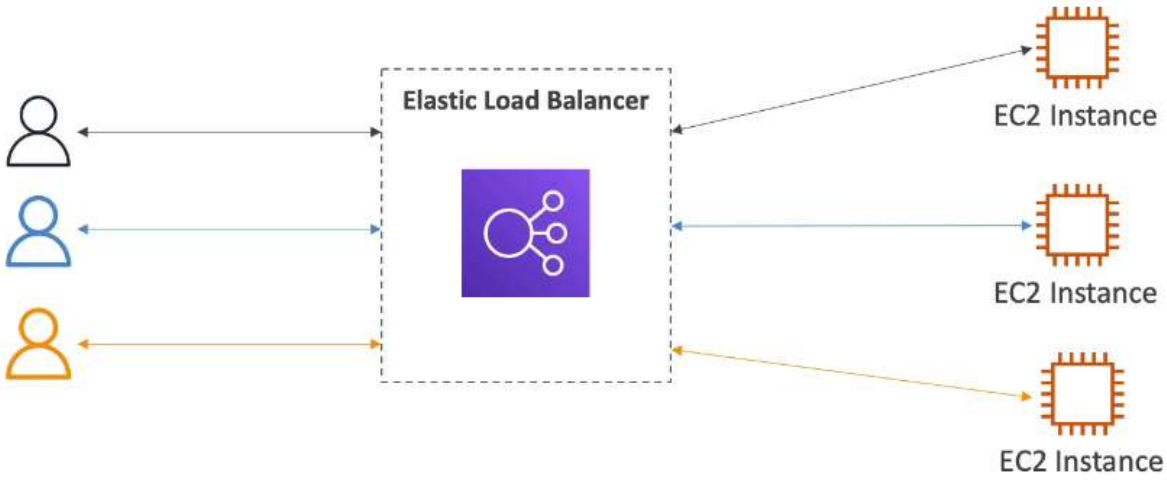
▼ ☑ **ELB.**

Load balancers are servers that forward internet traffic to multiple servers (EC2 Instances) downstream.

Pros of using a load balancer:

- **High availability across zones.**
- Spread load across multiple downstream instances.
- Expose a single point of access (DNS) to our application.
- Handle failures of downstream instances.
- **Do regular health checks to our instances**.
- Provide SSL termination for our websites easily.

ELB is a managed load balancer. AWS guarantees that will be working, takes care of upgrades, maintenance and high availability. ELB captures the logs and stores them in the S3 bucket that we specify as compressed files. We can disable access logging at any time.



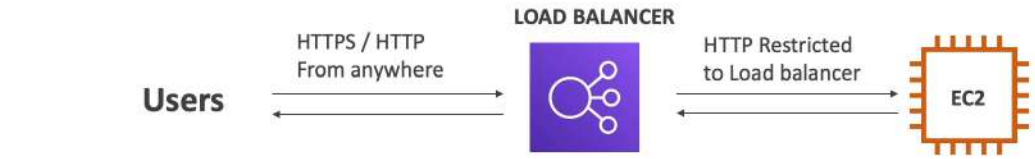
Health checks are crucial for Load Balancers. They enable the load balancer to know if instances it forwards traffict to are available to reply to requests. The health check is done on a port and a route (/health is common).

There are 3 types of load balancers offered by AWS:

- **Application Load Balancer (ALB)** (HTTP/HTTPS only) - Layer 7
- **Network Load Balancer (NLB)** (High performance, allows for TCP/UDP) - Layer 4
- **Gateway Load Balancer (GWLB)** - Layer 3

Some load balancers can be setup as interal (private) or external (public) ELBs.

ELB Security Groups



Load Balancer Security Group:

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	Allow HTTP from an...
HTTPS	TCP	443	0.0.0.0/0	Allow HTTPS from a...

Application Security Group: Allow traffic only from Load Balancer

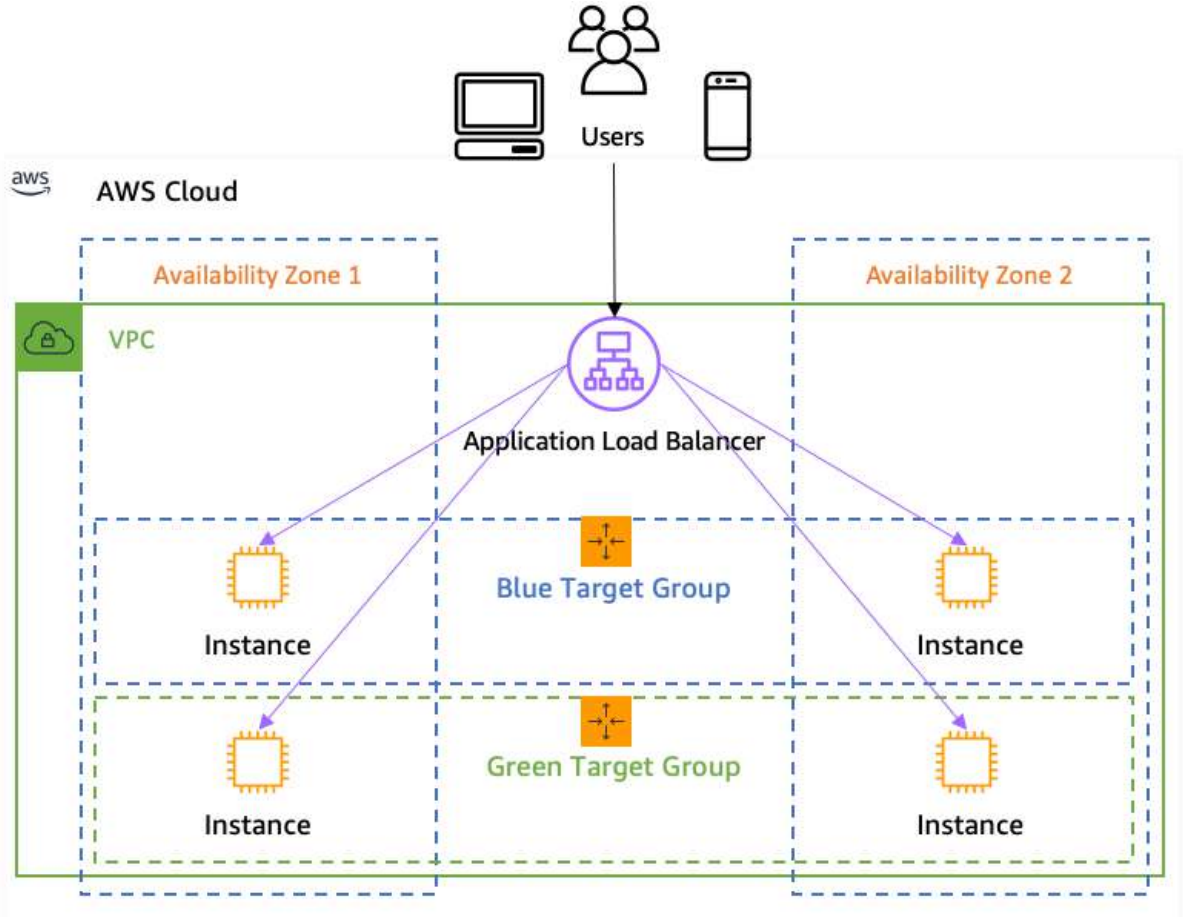
Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	sg-054b5ff5ea02f2b6e (load-b	Allow Traffic only...

▼ ☑ **Application Load Balancer (ALB).**

ALB operates at the application layer (Layer 7) of the OSI model, which enables it to make routing decisions based on content. It supports redirects (for example from HTTP to HTTPS).

ALB supports target groups, which are logical groups of resources. Each target group can have different health checks and configurations, allowing for fine-grained control. ALB can route to multiple target groups and health checks are at the target group level.

Slow Start Mode - allows us to add new targets without overwhelming them with a flood of requests.

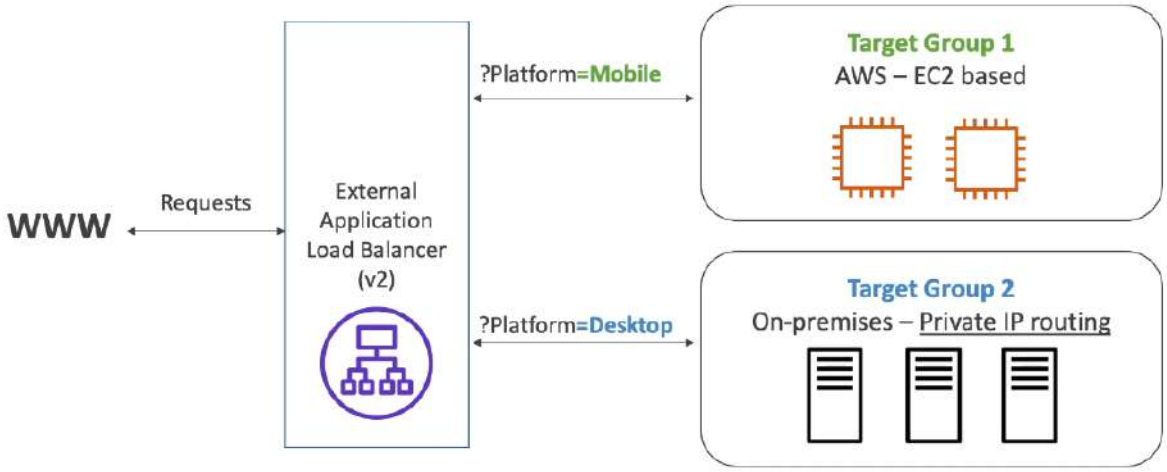


Target Groups:

- **EC2 Instances** (can be managed by an ASG).
- **ECS tasks** (managed by ECS).

- **Lambda functions**.
- **IP Addresses** (must be private IPs).

ALBs are a great fit for microservices & container-based applications. They have a port mapping feature to redirect to a dynamic port in ECS.



Routing Features

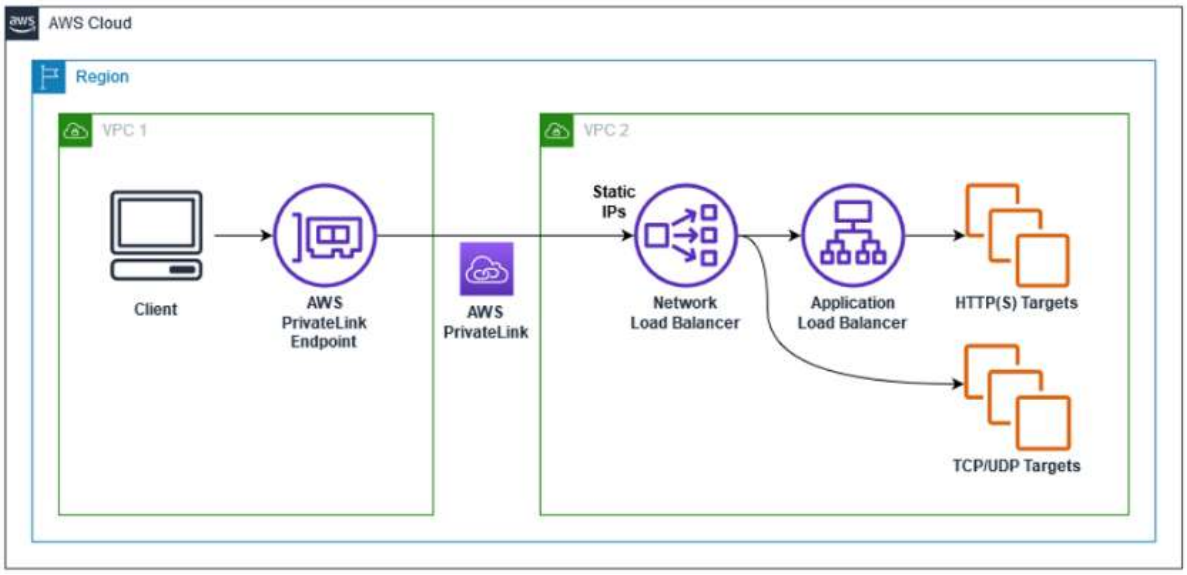
- **Path-based Routing** - based on the URL path (example.com/users).
- **Host-based Routing** - based on the Host field in the HTTP header (one.example.com).
- **Query String and Header-based Routing** - ased on the query string or specific headers in the HTTP request (exampe.com/users?id=123&order=false).

The application servers don't see the IP of the client directly. The true IP of the client is inserted in the header **X-Forwarder-For**.

▼ ☑ **Network Load Balancer (NLB).**

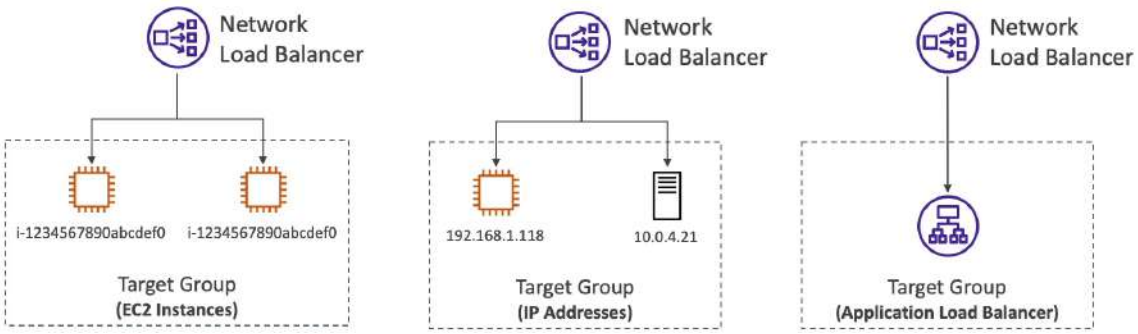
NLB operates at transport layer (Layer 4). Can handle high volumes of traffic with ultra-low latency, making it suitable for applications that require extreme performance, low latency, and TCP/UDP-level traffic handling. Network Load balancer doesn't have Weighted Target Groups.

NLB has **one static IP per AZ** and supports assigning Elastic IP (helpful for whitelisting specific IP).



Target Groups:

- **EC2 Instances**.
- **IP Addresses** (must be private IPs).
- **ALB**.



Health checks support the TCP, HTTP and HTTPS protocols.

If we specify targets using an instance ID, traffic is routed to instances using the primary private IP address specified in the primary network interface for the instance.

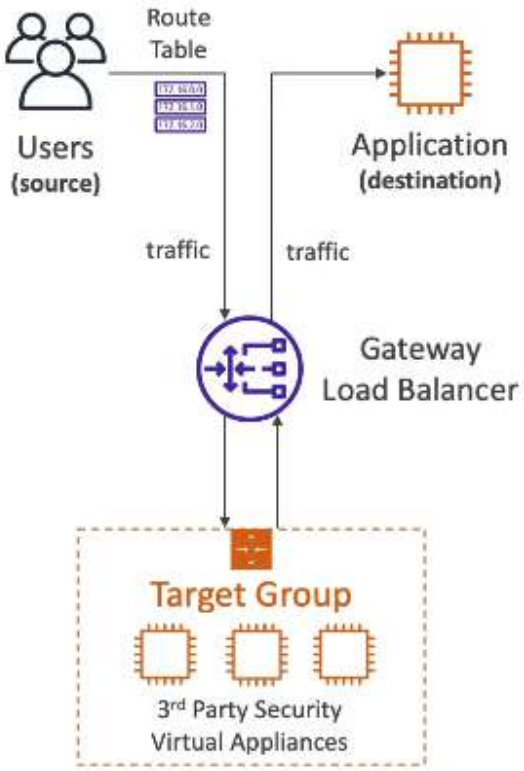
▼ ☑ **Gateway Load Balancer (GWLB).**

GWLB operates at network layer (Layer 3). It routes traffic based on IP addresses and supports both IPv4 and IPv6. This makes it highly efficient for managing network traffic and distributing it across multiple network appliances.

GLWB combines following functions:

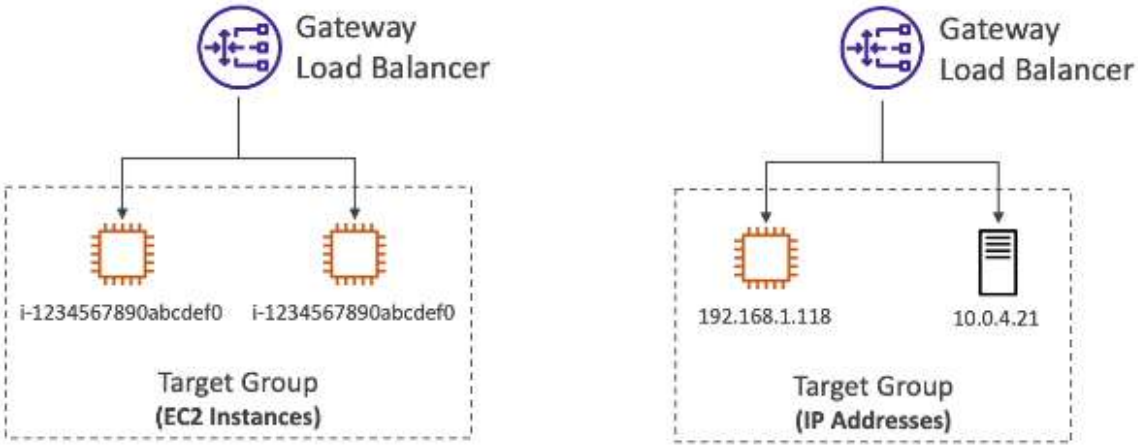
- Transparent Network Gateway - routes traffic without changing the source or destination IP addresses (**bump-in-the-wire**). It provides a single point of entry for all network traffic.
- Load Balancer - distributes traffic to our virtual appliances.

It uses the GENEVE protoctol on port 6081.



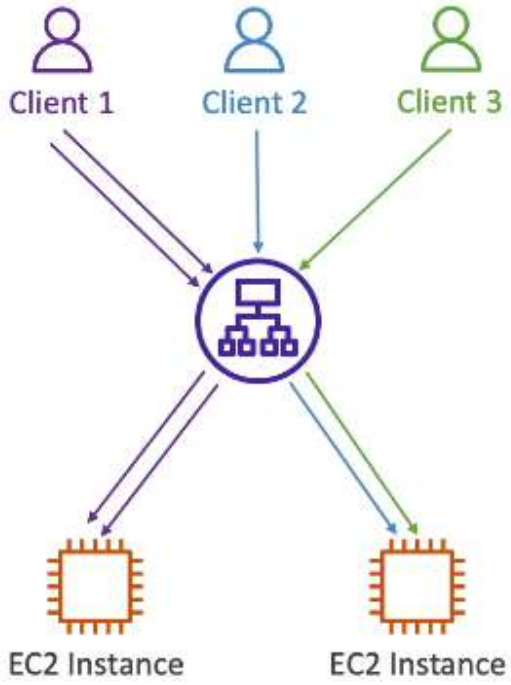
Target Groups:

- **EC2 Instances**.
- **IP Addresses** (must be private IPs).



▼ ☑ **ELB Sticky Sessions (Session Affinity).**

It is possible to implement stickiness so that the same client is always redirected to the same instance behind a load balancer. This works for ALB and NLB. It is used to prevent users from losing their session data.



Enabling stickinesss may bring imbalance to the load over the backend EC2 instances.

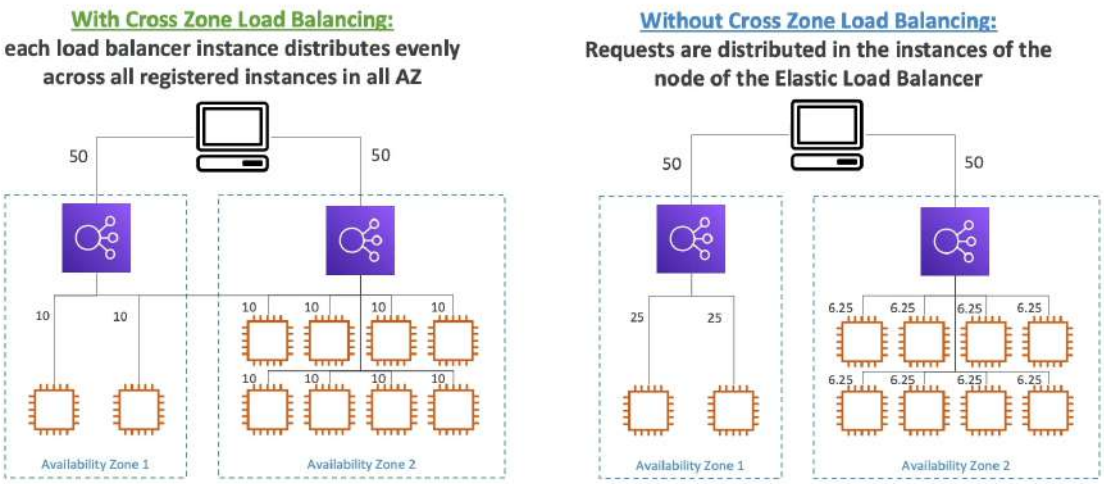
For ALB the cookie is used for stickiness. There are two types of cookies:

- **Application-based Cookies**

    - **Custom Cookie** - it is generated by the target. Can include any custom attributes required by the application. Cookie name must be specified individually for each target group (cannot use names AWSALB, AWSALBAPP or AWSALBTG).
    - **Application Cookie** - is is generated by the load balancer and cookie name is AWSALBAPP.
- **Duration-based Cookies** - it is generated by the load balancer. Cookie name is AWSALB.

▼ ☑ **ELB Cross-Zone Load Balancing.**

Cross-Zone Load Balancing is a feature ELB that helps distribute incoming traffic more evenly across the registered instances in different AZs.



- **ALB**: Cross-Zone Load Balancing is enabled by default and can be disabled at the target group level. We dont pay for inter AZ data.
- **NLB**: It is disabled by default and can be enabled via the AWS Management Console, CLI, or API. We need to pay charges for inter AZ data if enabled.

▼ ☑ **ELB SSL Certificates.**

SSL Certificate allows traffic between our clients and our load balancer to be encrypted in transit. They have an expiration date and must be renewed. Public SSL certificates are issued by CA (Certificate Authorities).

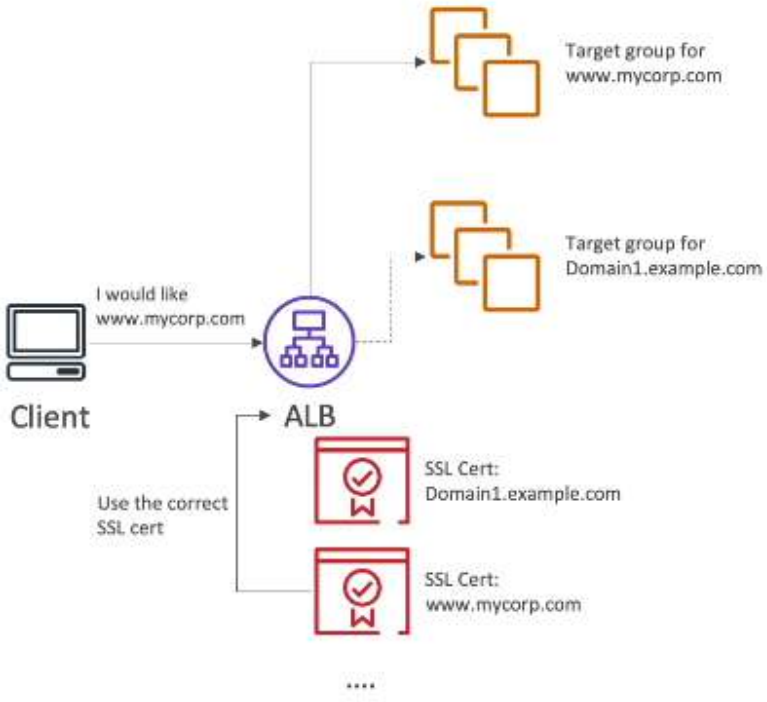
- SSL (Secure Sockets Layer) - used to encrypt connections.

- TLS (Transport Layer Security) - successor to SSL.

We can manage certificates using ACM and can create and upload our own certificates alternatively. The load balancer uses an X.509 certificate (SSL/TLS server certificate).

HTTPS Listener - a configuration that enables the load balancer to handle HTTPS. We must specify a default certificate and can add an optional list of certificates to support multiple domains. Clients can use **SNI (Server Name Indication)** to specify the hostname they reach. Also we have ability to specify a security policy to support older versions of SSL/TLS (legacy clients).

Server Name Indication (SNI) - solves the problem of loading multiple SSL certificates onto one web server (to serve multiple websites). It is a newer protocol and requires the client to indicate the hostname of the target server in the initial SSL handshake. It only works with **ALB, NLB & CloudFront**.

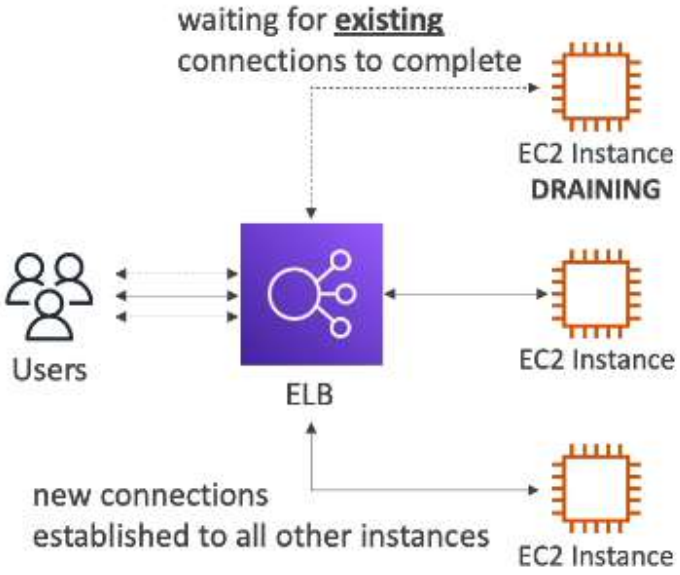


ALB & NLB support multiple listeners with multiple SSL certificates and use SNI to make it work.

▼ ☑ ELB Deregistration Delay (Connection Draining).

Deregistration Delay ensures that in-flight requests are handled properly when instances are deregistered or terminated. It stops sending new requests to the EC2 instance which is de-registering.

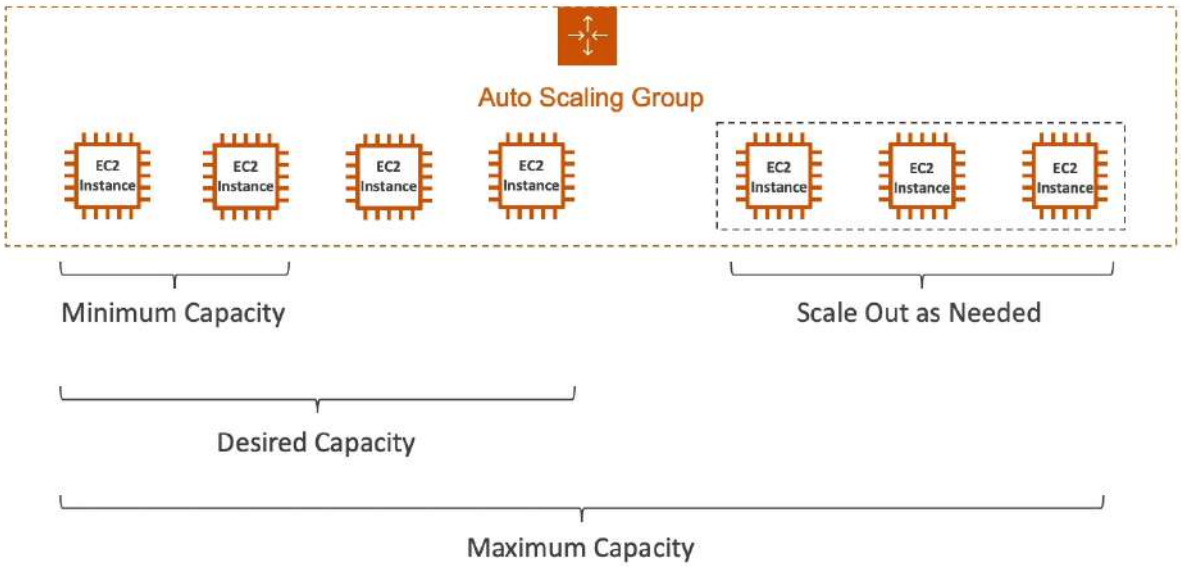
We can set timeout between 1 to 3600s (default is 300s). It can be disabled if we se value to 0. Good practice is to set to a low value if our requests are short.



▼ ☑ ASG.

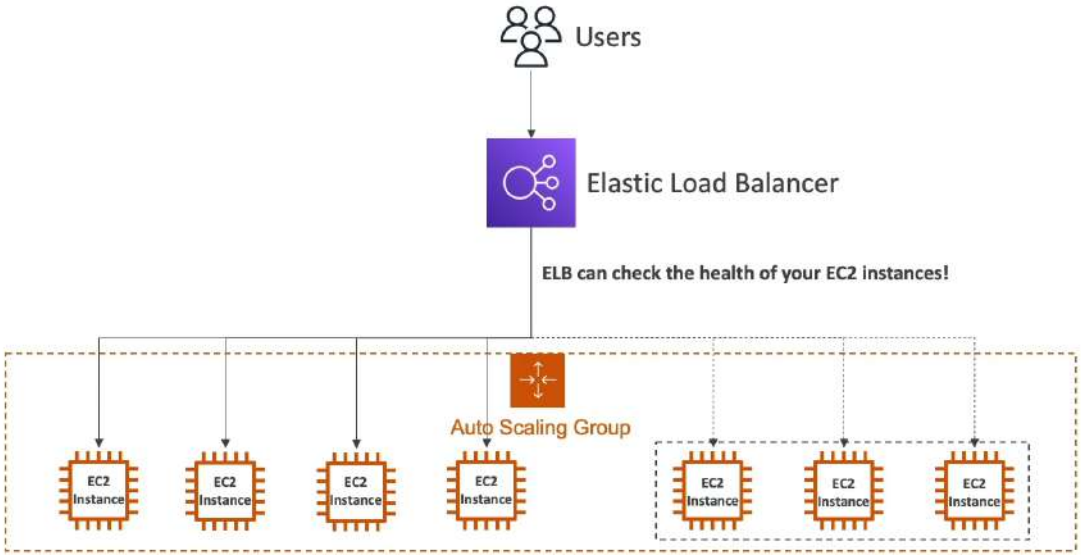
The goal of **ASG** is to:

- Scale out (add EC2 instances) to match an increased load.
- Scale in (remove EC2 instances) to match a decreased load.
- Ensure we have a minimum and a maximum number of machines running.
- **Automatically register new instaces to a load balancer.**
- **Replace unhealthy instances**.



When EC2 Auto Scaling responds to a **scale-out** event, it launches one or more instances. These instances start in the Pending state. If we added an **autoscaling:EC2_INSTANCE_LAUNCHING** lifecycle hook to our ASG, the instances move from the **Pending** state to the **Pending:Wait** state. After we complete the lifecycle action, the instances enter the **Pending:Proceed** state. When the instances are fully configured, they are attached to the ASG and they enter the **InService** state.

When Amazon EC2 Auto Scaling responds to a **scale-in** event, it terminates one or more instances. These instances are detached from the ASG and enter the **Terminating** state. If we added an **autoscaling:EC2_INSTANCE_TERMINATING** lifecycle hook to our ASG the instances move from the **Terminating** state to the **Terminating:Wait** state. After we complete the lifecycle action, the instances enter the **Terminating:Proceed** state. When the instances are fully terminated, they enter the **Terminated** state.



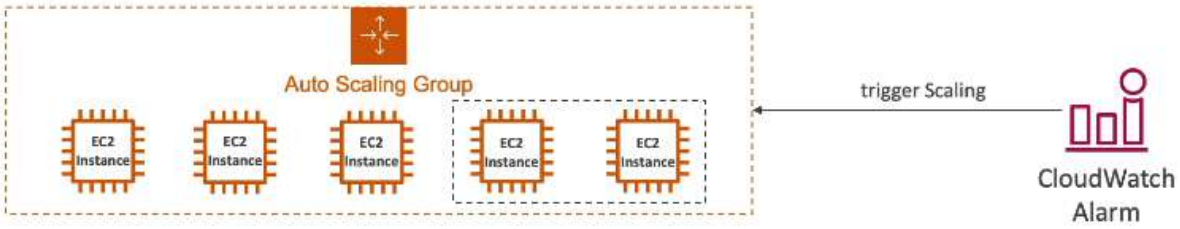
ASG Attributes

- **Launch Template**

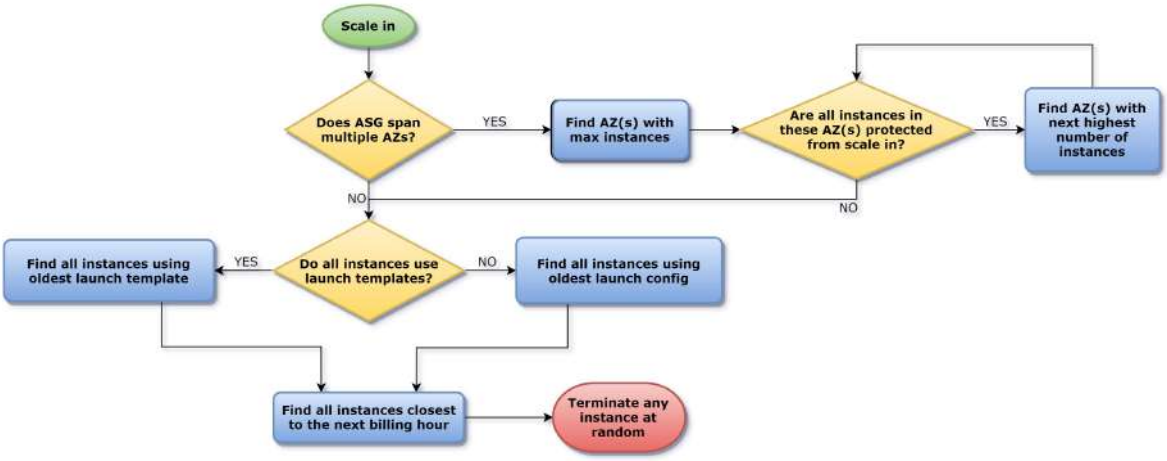
  It acts as a blueprint for EC2 instances, allowing us to define a set of configuration parameters that can be reused when we launch new instances. When instances are launched via a launch template, they can automatically be registered with an ELB. Once we have created a launch template, it can't be modified.

- **Min Size / Max Size / Initial Capacity**
- **Scaling Policies**

It is possible to scale an ASG based on CloudWatch alarms. Based on the alarm we can create scale-out policies or scale-in policies.



EC2 Termination Policy



▼ ☑ **ASG Scaling Policies.**

- **Manual Scaling** - Update the size of an ASG manually. Suitable for predictable workloads where we can manually scale based on known traffic patterns.
- **Dynamic Scaling** - Respond to changing demand.
    - **Simple/Step Scaling**

      Useful for applications with variable demand where we need more granular control over scaling actions.

      Example: When a CloudWatch alarm is triggered (example CPU > 70%), then add more units. When CloudWatch alarm is triggered (example CPU

< 30%), then remove unit.

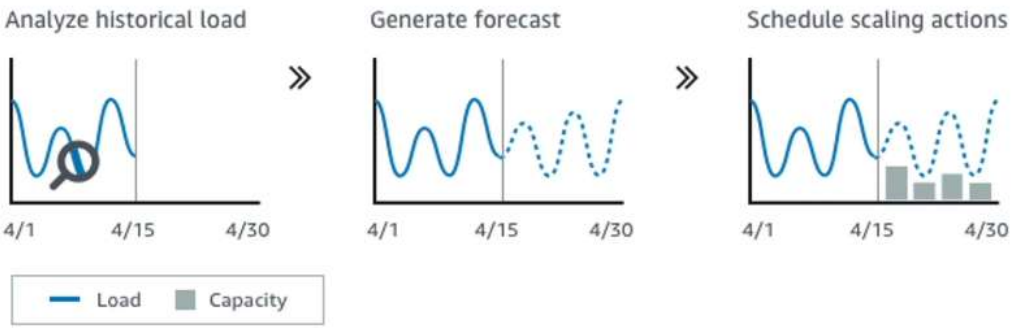
- **Target Tracking Scaling**

  Simplifies scaling by allowing us to set a desired target value for a specific metric, making it ideal for applications with varying demand and performance requirements. Example: If we want the average ASG CPU to stay at around 40%.

- **Scheduled Scaling**

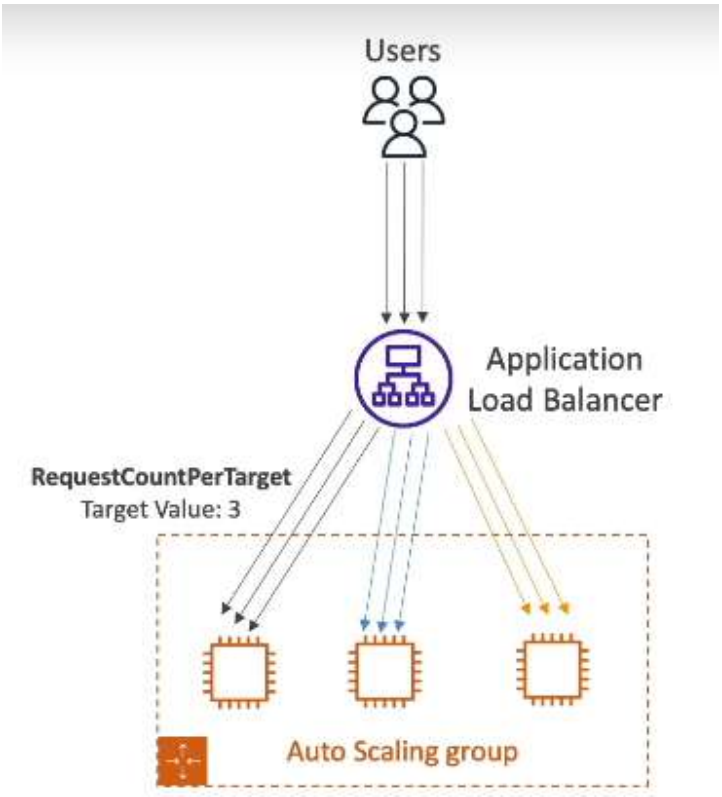
  Ideal for applications with predictable traffic patterns, such as regular business hours or batch processing jobs. Example: Increase the min capacity to 10 at 5PM on Fridays.

- **Predictive Scaling** - Uses ML to predict future traffic ahead of time. Automatically provisions the right number of EC2 instances in advance. Suitable for applications with periodic, predictable load changes, allowing us to proactively scale and avoid latency or over-provisioning.



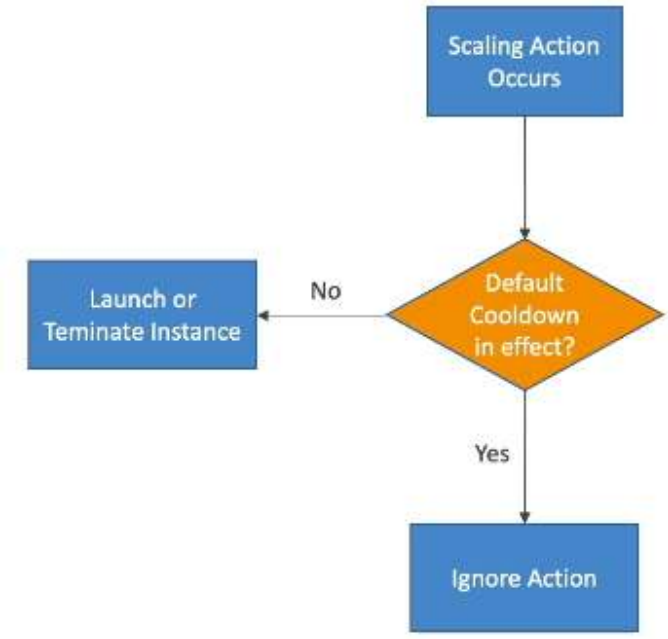
Good metrics to scale on:

- **CPU Utilization** (average CPU utilization across our instances).
- **Request Count Per Target**.



- **Average Network In/Out**.
- **Any custom metric** (that we pust using CloudWatch).

Scaling Cooldown - after a scaling activity happens, we are in the cooldown period (default 300s). During the cooldown period, the ASG will not launch or terminate additional instances (to allow for metrics to stabilize).



Good practice is to use a ready-to-use AMI to reduce configuration time in order to be serving request faster and reduce the cooldown period.

RDS, Aurora & ElastiCache

▼ ☑ **RDS.**

RDS is a managed DB service which uses SQL as a query language. It allows us to create databases in the cloud that are managed by AWS (Postgres, MySQL, MariaDB, Oracle, Microsoft SQL Server, Aurora).

RDS is a managed service:

- Automated provisioning, OS patching.
- Continous backups and restore to a specific timestamp (Point in Time Restore).
- Dashboards monitoring.
- Read replicas for improved read performance.
- Multi AZ setup for disaster recovery.
- Maintenance windows for upgrades.

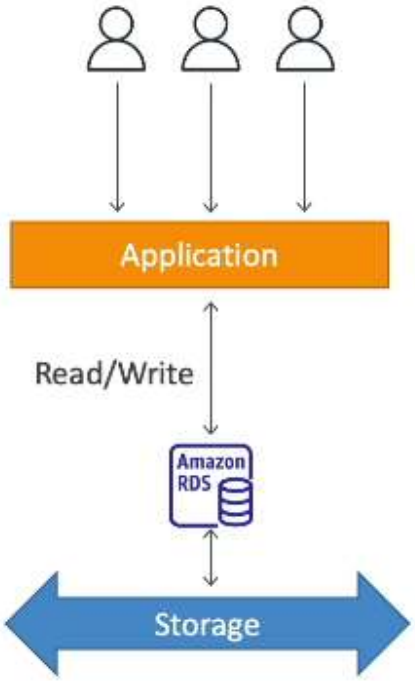
- Vertical and horizontal scaling capability.
- Storage backed by EBS.

We cant SSH into our RDS instances.

RDS Storage Auto Scaling helps us increase storage on our RDS instance dynamically. When RDS detects we are running out of free database storage, it scales automatically. We have to set maximum storage treshold. It is useful for applications with unpredictable workloads.

It automatically modifies storage if:

- Free storage is less than 10% of allocated storage.
- Low-storage lasts at least 5 minutes.
- 6 hours have passed since last modifications.



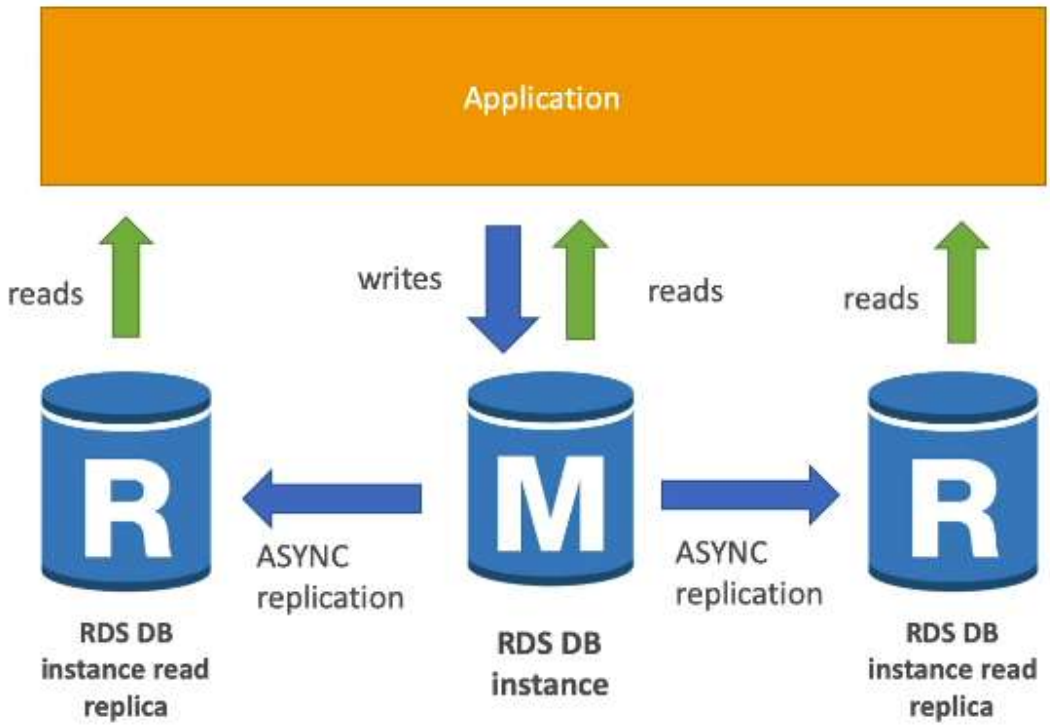
▼ ☑ **RDS Deployments.**

Read Replicas

Read Replicas are a feature that allows us to create one or more read-only copies (max 15) of our primary database instance. They can be within AZ, cross

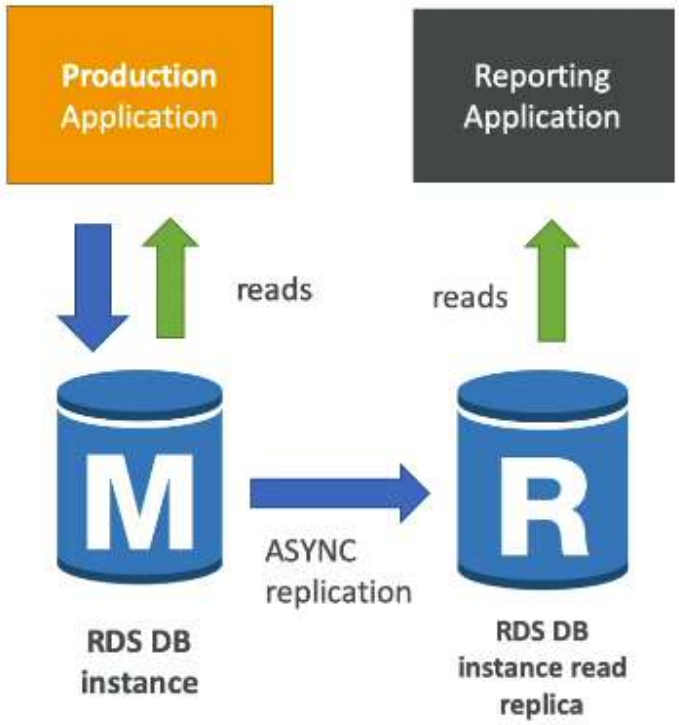
AZ or cross region. Applications must update the connection string to leverage read replicas.

Replication is **async**, so reads are eventually consistent. Replicas can be promoted to their own database.



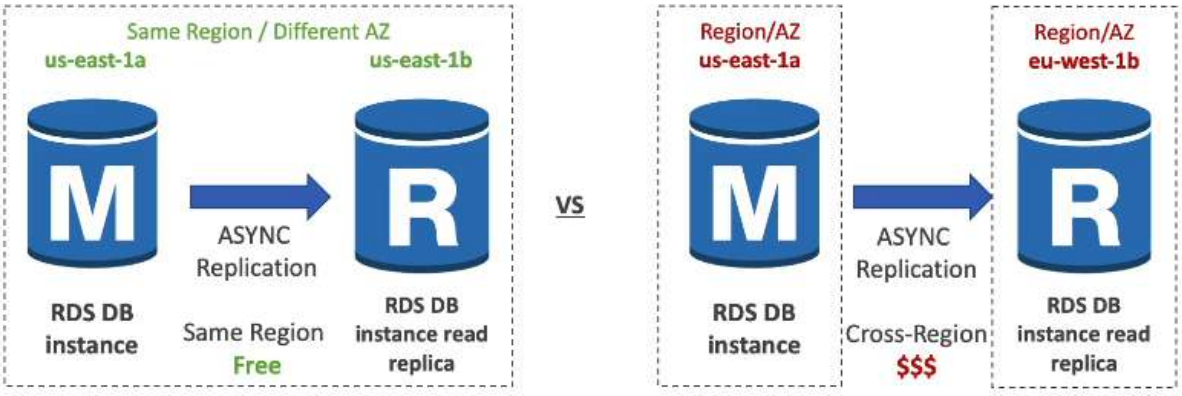
Read Replicas Use Cases:

- Having a production database that is taking on normal load.
- Want to run a reporting application to run some analytics.
- To run the new workload.
- To not affect production application.



Read replicas are used only for **SELECT** statements in queries.

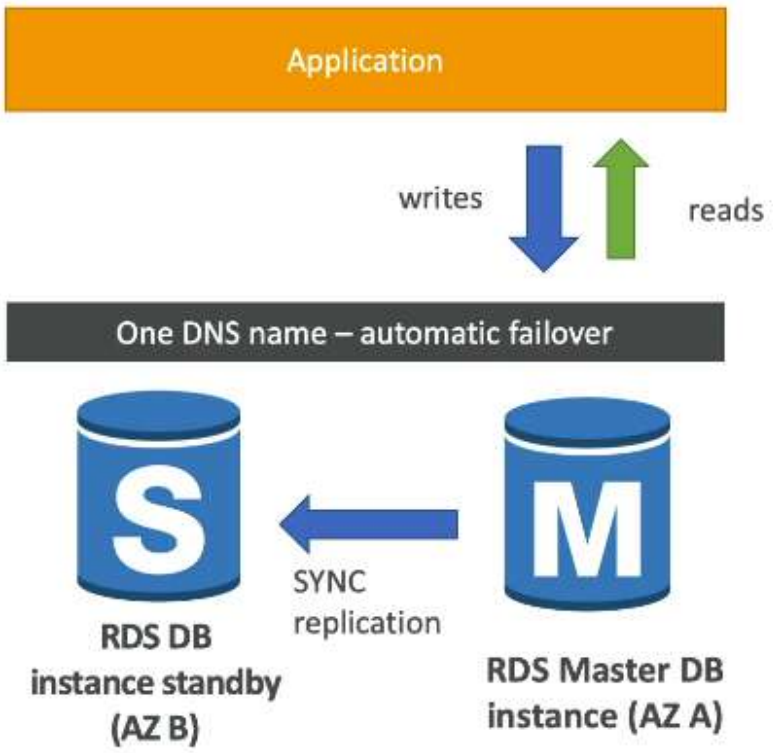
In AWS there is a network cost when data goes from one AZ to another. For RDS Read Replicas within the same region, we dont pay that fee.



Multi-AZ

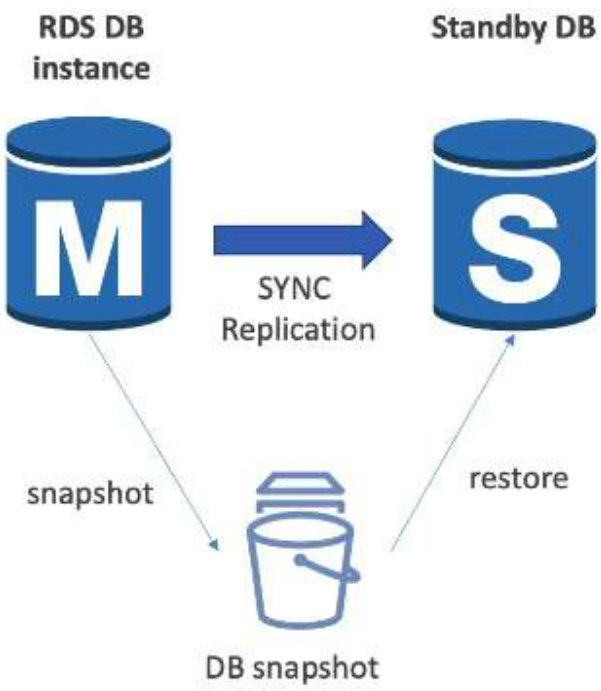
Multi-AZ deployments are designed to enhance the availability and durability of our database instances. This replication is **sync**. It is used for failover in case of loss of AZ, loss of network, isntace or storage failure. Not used for scaling.

> Read replicas can be set too as Multi-AZ for DR.



It is possible to move RDS database from Single-AZ to Multi-AZ. It is zero downtime operation.

Snapshot of database will be taken. Then new database is restored from the snapshot in a new AZ. After that sync is established between the two databases.



Multi-AZ DB Cluster Deployments - high availability and durability by replicating data across multiple AZs and supporting automatic failover at both the instance and cluster levels.

▼ ☑ **RDS Custom for Oracle & Microsoft SQL Server.**

RDS Custom is a feature that provides a way to use a custom database engine. It's used for Oracle and Microsoft SQL Server with OS and database customization (**access to the underlying database and OS**).

We can configure settings, install patches, access underlying EC2 using SSH or SSM.

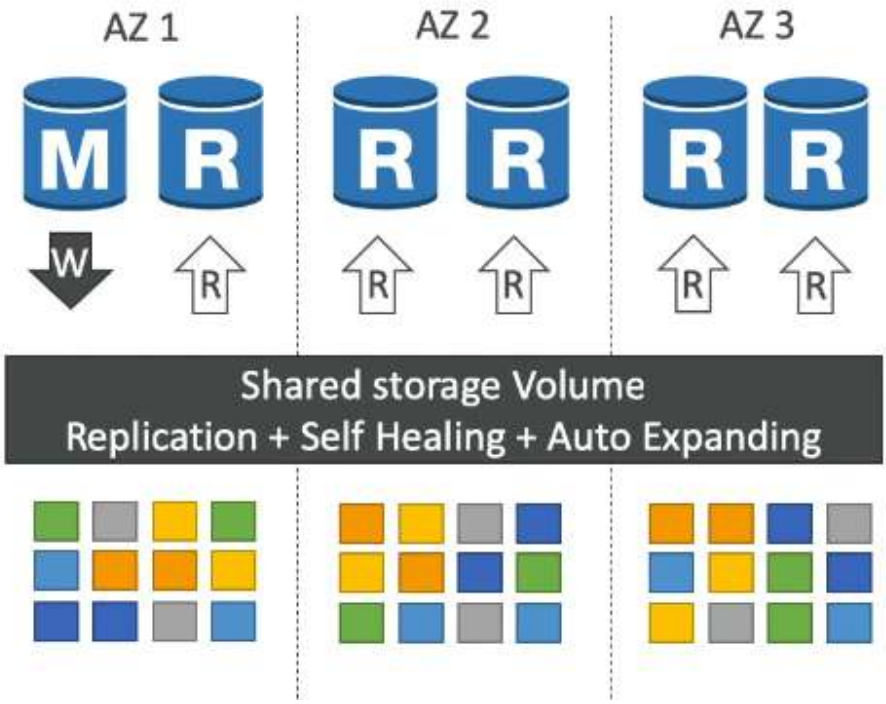
To perform our customization it is recommended deactivate Automation Mode (better to take a DB snapshot before).

▼ ☑ **Aurora.**

Aurora is a properietary technology from AWS. It supports Postgres and MySQL. Aurora has 3x performance improvement over Postgres and 5x performance improvement over MySQL. Aurora storage automatically grows in increments of 10GB, up to 128TB.

Aurora stores 6 copies of our data across 3 AZ (4 copies out of 6 are needed for writes and 3 copies out of 6 are needed for reads). It **has self healing with peer-to-peer replication**.

Only one Aurora instance takes writes (**master**). Automated failover for master is less than 30s. We can have up to 15 Aurora read replicas for reads and they all support CRR.

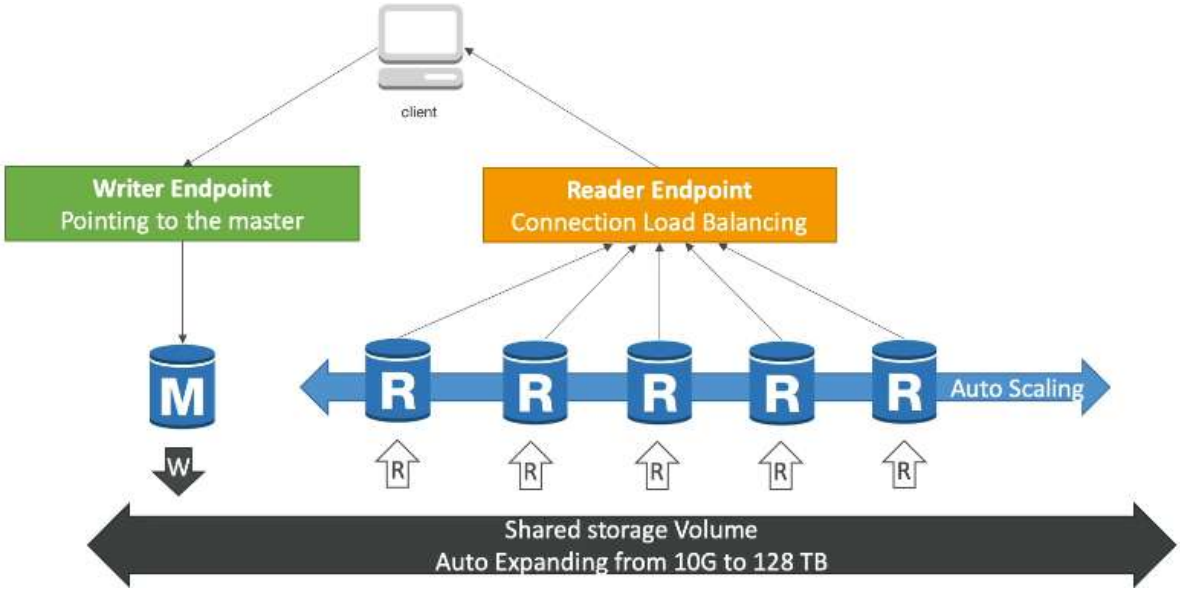


If we have an Aurora Replica in the same or a different AZ, when failing over, Aurora flips the canonical name record (CNAME) for our DB Instance to point at the healthy replica, which in turn is promoted to become the new primary. Start-to-finish failover typically completes within 30 seconds.

Aurora Endpoints

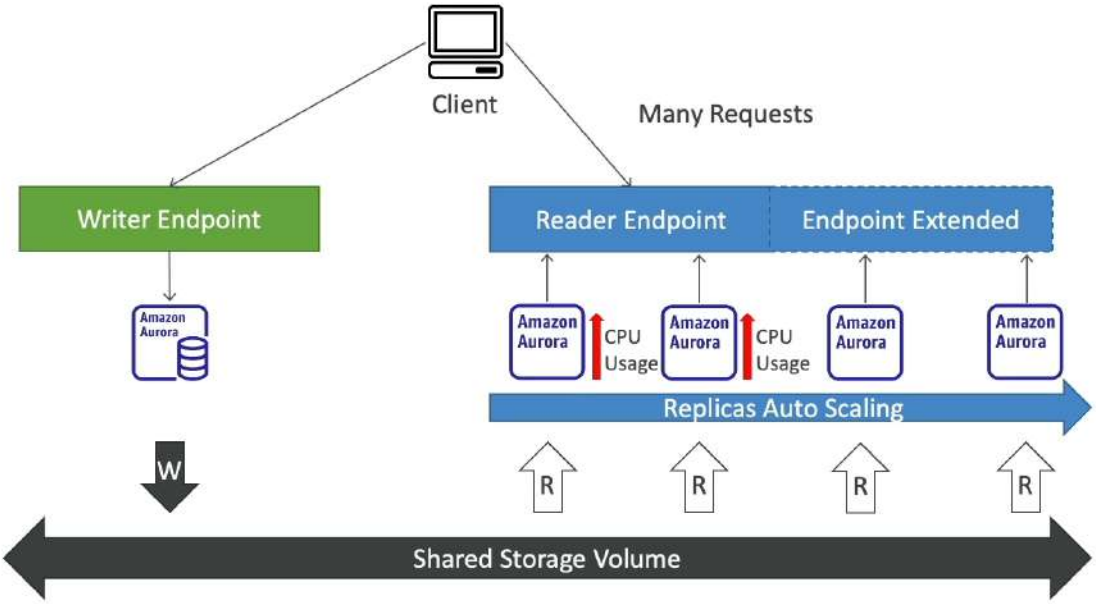
Writer Endpoint - used for all write operations to the Aurora database. This endpoint directs traffic to the primary instance within the Aurora cluster. In the event of a failure of the master instance, Aurora automatically promotes one of the replicas to be the new primary instance.

Reader Endpoint - allows applications to distribute read traffic across multiple Aurora replicas within the cluster, enhancing read scalability and performance. The reader endpoint automatically balances the read workload across all available replicas.

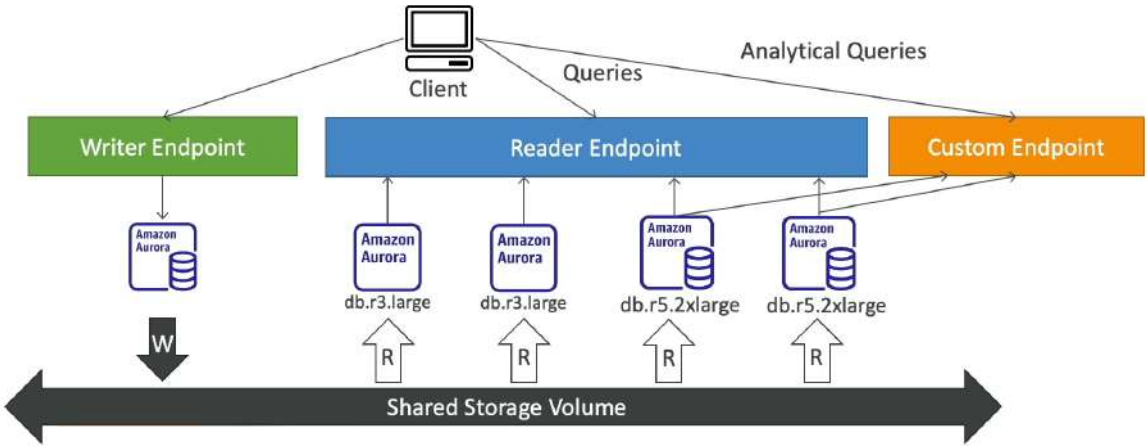


▼ ☑ **Aurora Advanced Concepts.**

Aurora Replicas Auto Scaling - automatically adjusts the number of Aurora Replicas in response to changes in our application's workload.



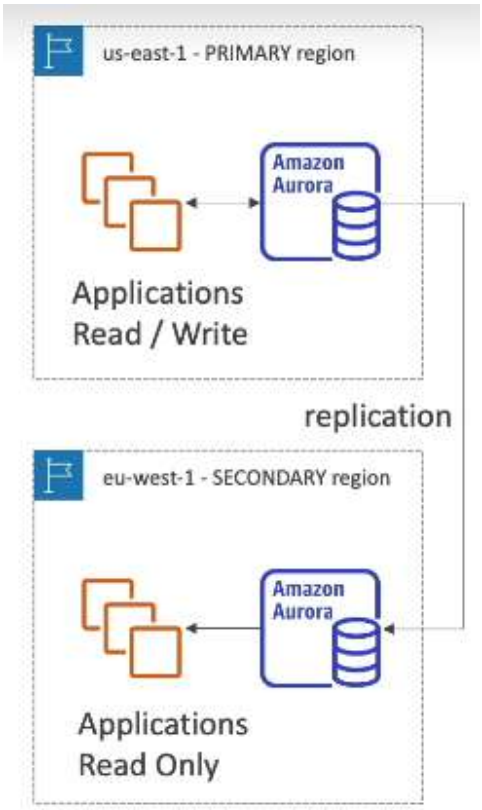
Aurora Custom Endpoints - allow us to define and configure additional endpoints for our Aurora DB cluster, beyond the default writer and reader endpoints.



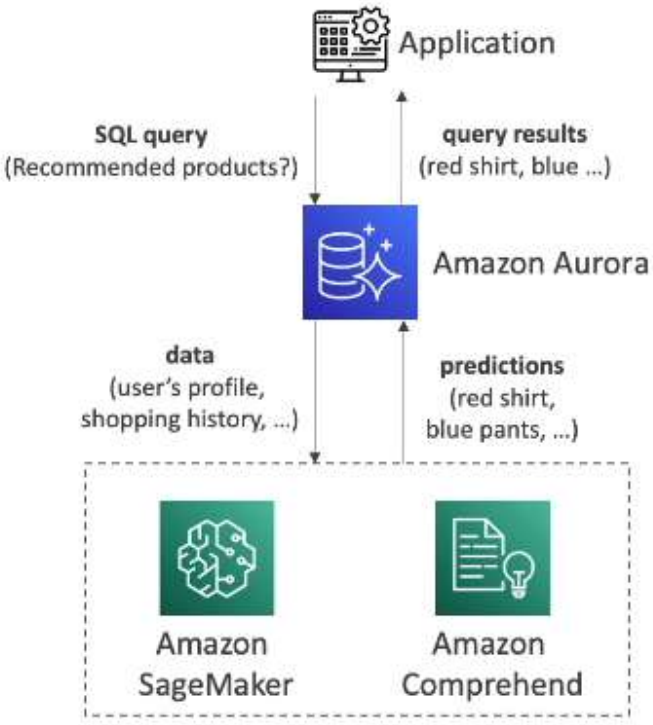
Aurora Serverless - automated database instantiation and auto-scaling based on actual usage (no management). No capacity planning is needed for it and we pay per second (can be more cost-effective). Can be used for infrequent or unpredictable workloads.

Aurora Global - enables a single Aurora database to span multiple AWS regions, providing low-latency global reads, fast disaster recovery, and cross-region data replication.

There is 1 primary region for read and write and up to 5 secondary regions for read-only. We can set up to 16 read replicas per secondary region. Typical cross-region replication takes less than 1s.



Aurora Machine Learning - enables us to add ML-based predictions to our applications via SQL. We don't need to have ML experience. It is used for fraud detection, ads targeting, sentiment analysis, product recommendations...



▼ ☑ **RDS & Aurora Backup & Monitoring.**

RDS Enhanced Monitoring - provides detailed **real-time** insights into the OS metrics of our RDS instances. It helps us monitor the performance of our RDS instances more effectively than basic monitoring alone.

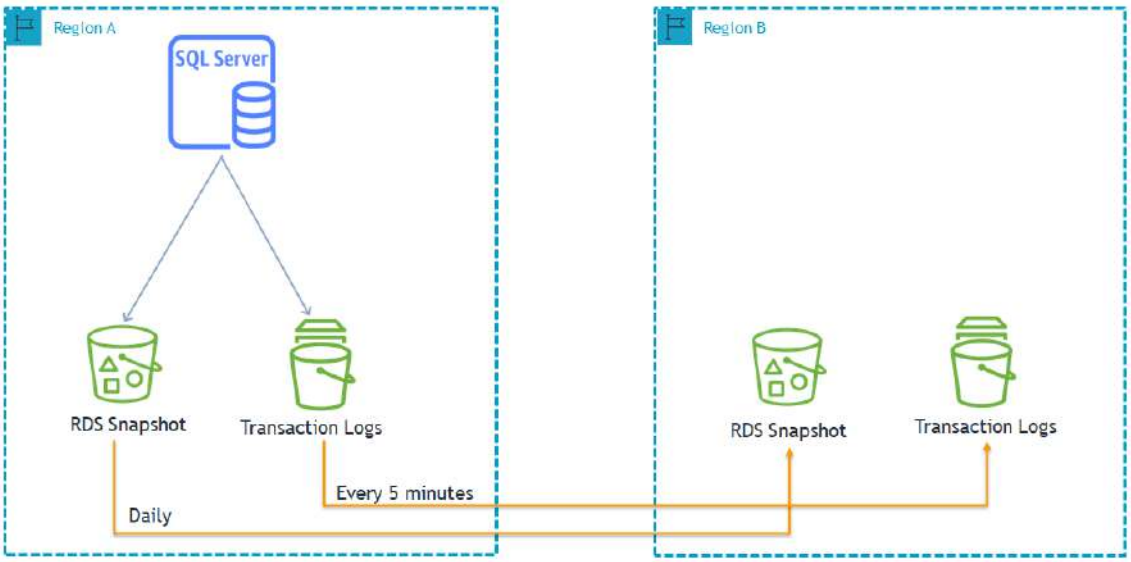
RDS Performance Insights is a feature that helps us monitor and analyze the performance of our RDS databases. It is offering a higher-level view of how our database queries and workloads are performing over time.

RDS Backups

- **Automated backups**
  - Daily full backup of database (during the backup window).
  - Transaction logs are backed-up by RDS every 5mins.

  With automated backups we have ability to restore to any point in time (from oldest backup to 5mins ago).

  Retention of backups can be set from 1 to 35 days (set to 0 to disable automated backups.)



- **Manual DB Snapshots**

  They are manually triggered by the user. Retention of backups can be set for as long as needed.

> In a stopped RDS database, we will still pay for storage. **If we plan on stopping it for a long time, we should use snapshot & restore instead.**

Aurora Backups

- **Automated backups** - from 1 to 35 days (cannot be disabled). We have point-in-time recovery in that timeframe.
- **Manual DB Snapshots** - manually triggered by the user. Retention of backups can be set for as long as needed.

RDS & Aurora Restore Options

- **Restoring RDS/Aurora backup or snapshot** - creates a new database.
- **Restoring MySQL RDS database from S3**
  1. Create a backup of our on-premises database.
  2. Store it on S3.
  3. Restore the backup file onto a new RDS instance running MySQL.
- **Restoring MySQL Aurora cluster from S3**

1. Create a backup of our on-premises database using Percona XtraBackup.
2. Store the backup file on S3.
3. Restore the backup file onto a new Aurora cluster running MySQL.

Aurora Database Cloning

It is possible to create a new Aurora DB Cluster from an existing one. It is cost effective and faster than snapshot & restore.

It uses copy-on-write protocol - initially, the new database cluster uses the same data volume as the original database cluster (no copying needed). When updates are made to the new database cluster data, then additional storage is allocated and data is copied to be separated.

Cloning is useful to create a "staging" database from a "production" database without impacting the production database.

▼ ☑ **RDS & Aurora Security.**

- **At-rest encryption** - database master & replicas encryption using KMS (must be defined as launch time). If the master is not encrypted, the read replicas cannot be encrypted. To encrypt an un-encrypted database we should go through a database snapshost & restore as encrypted.
- **In-flight encryption** - they are TLS-ready by default and use the TLS root certificates client-side.
- **IAM Authentication** - we should use IAM roles to connect to our database. Instead of username and password, we use an authentication token. IAM database authentication works with MySQL and PostgreSQL. Network traffic to and from the database is encrypted using SSL.
- **Security Groups** - control network access to our databases.
- **No SSH available** (except on RDS Custom).
- **Audit Logs** - can be enabled and sent to CloudWatch Logs for longer retention.

▼ ☑ **RDS Proxy.**

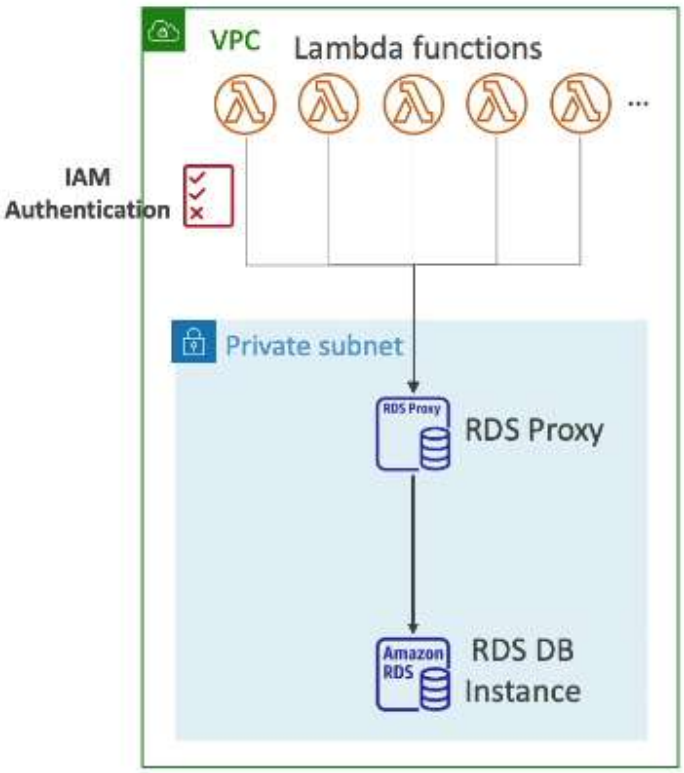
RDS Proxy is a fully managed (serverless, autoscaling, multi-AZ) database proxy for RDS. It allows apps to pool and share database connections established with

the database. No code changes are required for most apps to integrate with RDS Proxy.

It improves database efficiency by reducing the stress on database resources and minimizes open connections and timeouts. It reduces RDS & Aurora failover time by up to 66%.

RDS Proxy enforces IAM authentication for the database and securely stores credentials in SSM.

RDS Proxy is never publicly accessible (must be accessed from VPC).



▼ ☑ **ElastiCache.**

Amazon ElastiCache is a fully managed, in-memory caching service provided by AWS. ElastiCache supports two popular in-memory caching engines:

- **Redis** - in-memory data structure store that supports data types like strings, hashes, lists...
- **Memcached** - a high-performance, distributed memory object caching system. It is simple and ideal for use cases that require a large cache with

minimal operational overhead.

ElastiCache helps reduce load off databases for read intensive workloads. Using ElastiCache involves heavy application code changes.

ElastiCache Auto Discovery - the ability for client programs to automatically identify all of the nodes in a cache cluster, and to initiate and maintain connections to all of these nodes.

Redis vs Memcached

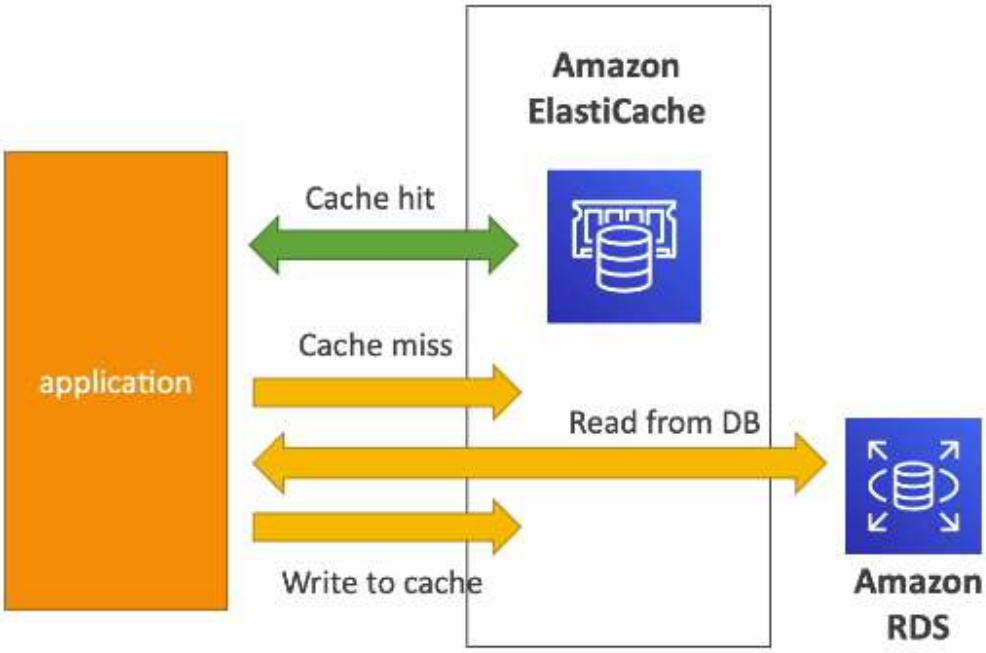
REDIS	MEMCACHED
• Multi AZ with Auto-Failover • Read Replicas to scale reads and have high availability • Data Durability using AOF persistence • Backup and restore features • Supports Sets and Sorted Sets	• Multi-node for partitioning of data (sharding) • No high availability (replication) • Non persistent • No backup and restore • Multi-threaded architecture

▼ **Solution Architecture - DB Cache.**

It is used to reduce the latency and load on a relational or NoSQL database by caching frequently accessed data.

Components:

- **Amazon ElastiCache (Redis/Memcached)**
- **Primary Database (RDS/DynamoDB)**
- **Application**



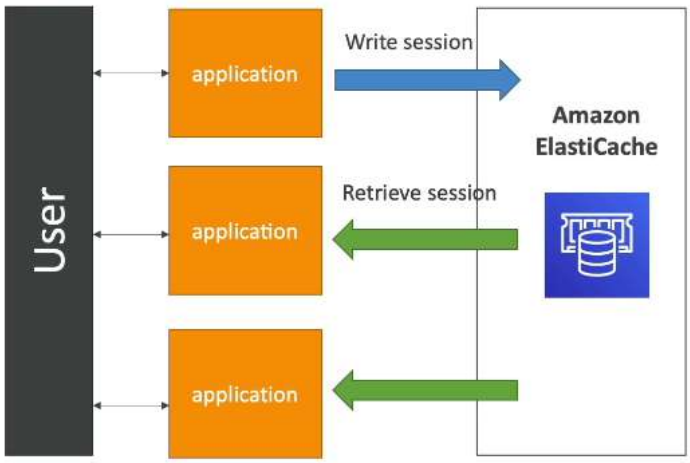
Cache must have an invalidation strategy to make sure only the most current data is used in there.

▼ **Solution Architecture - User Session Store.**

Used to manage user session data efficiently by storing it in an in-memory cache, providing fast access and improving application performance.

Components:

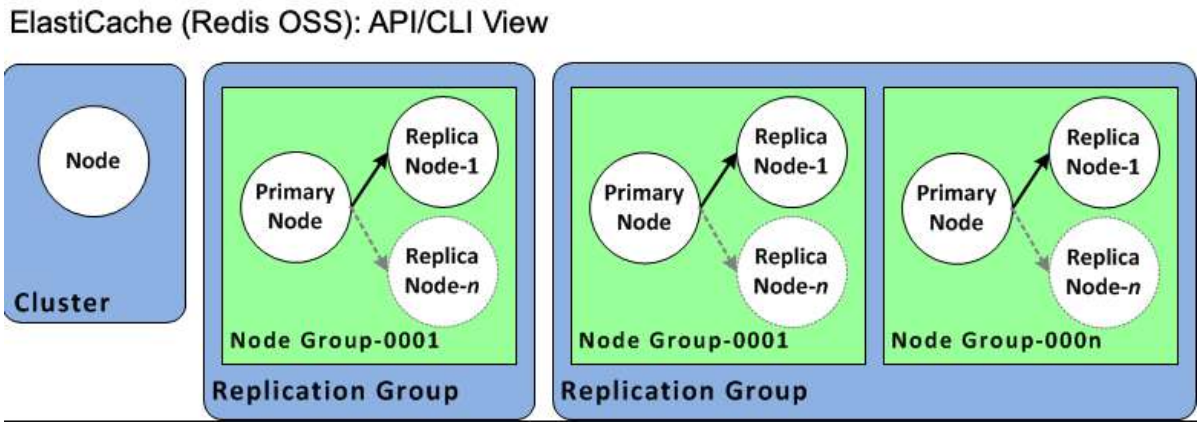
- **Amazon ElastiCache (Redis)**
- **Application Servers**
- **User Database (Optional)**



User logs into any of the application instances. The application writes the session data into ElastiCache. Then the user hits another instance of our application and instance retrieves the data and the user is already logged in.

▼ ☑ **ElastiCache Redis High Availability.**

Replication Groups are collections of Redis nodes (instances) where one is the **primary** node that handles writes, and the rest are **replica nodes** that handle reads and provide failover if the primary fails.



Shard in Redis is a subset of the dataset. In sharding, data is split across multiple Redis nodes, which helps with **horizontal scaling**. A shard can have a primary node and multiple replica nodes to provide redundancy and availability.

Redis supports a persistence mode called **Append-Only File (AOF)**. Every write operation is logged in an append-only file, ensuring that no data is lost in case of

failure. AOF provides **data durability** but may slightly degrade performance because of disk writes. Turning on AOF ensures **local** data persistence but may not help with **regional** failures. To provide resilience on region level we should use Multi-AZ replication groups.

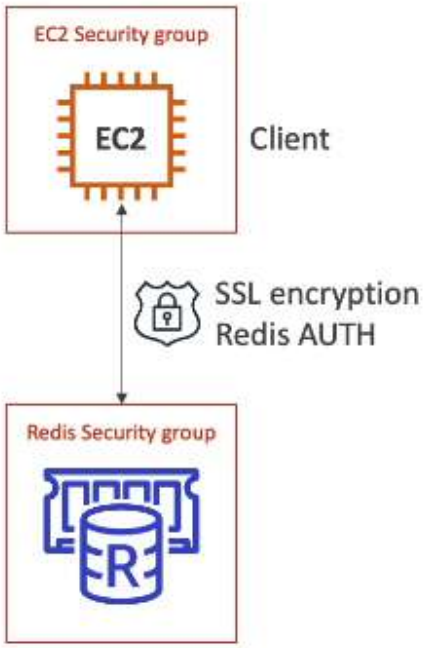
▼ ☑ **ElastiCache Advanced Concepts.**

Cache Security

ElastiCache supports IAM authentication for Redis. IAM policies on ElastiCache are only used for API-level security.

Redis AUTH - we can set a password/token when we create a Redis cluster. This is an extra level of security for our cache (on top of security groups). It supports SSL in-flight encryption.

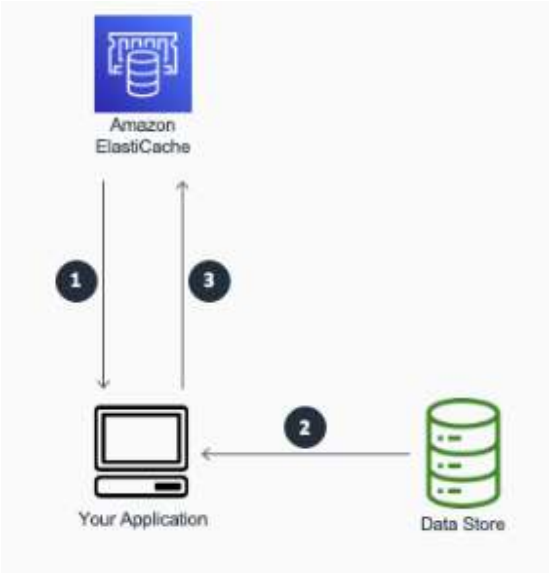
To require that users enter a password on a password-protected Redis server, include the parameter --auth-token with the correct password when we create our replication group or cluster and on all subsequent commands to the replication group or cluster.



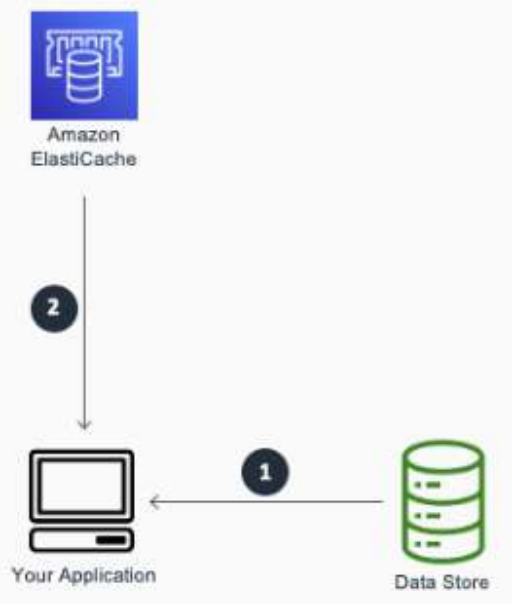
Memcached supports SASL-based authentication.

ElastiCache Design Patterns

- **Lazy Loading (Cache-Aside)** - all the read data is cached, data can become stale in cache.



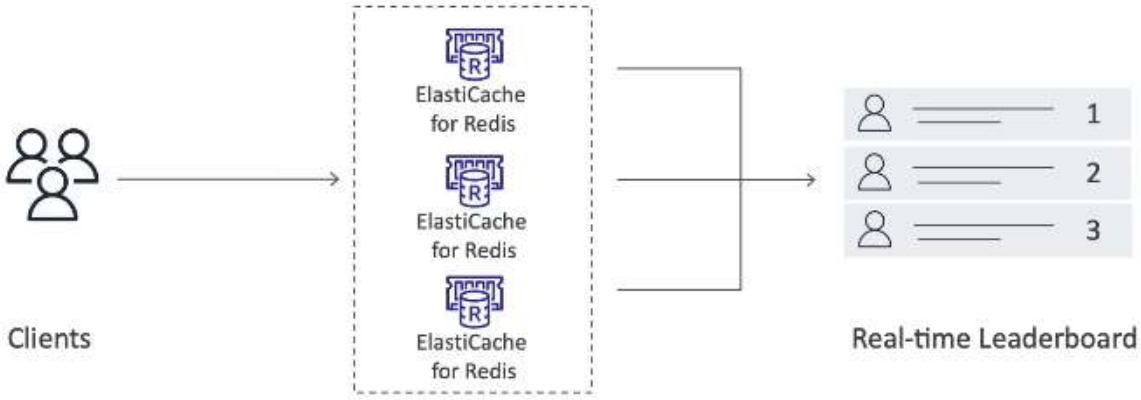
- **Write Through** - adds or updates data in the cache written to a database (no stale data).



- **Session Store** - store temporary session data in a cache (using TTL features).

ElastiCache - Redis Use Case

Gaming leaderboards are computationally complex and Redis sorted sets guarantee both uniqueness and element ordering to solve this problem. Each time a new element is added, it is ranked in real time, then added in correct order.



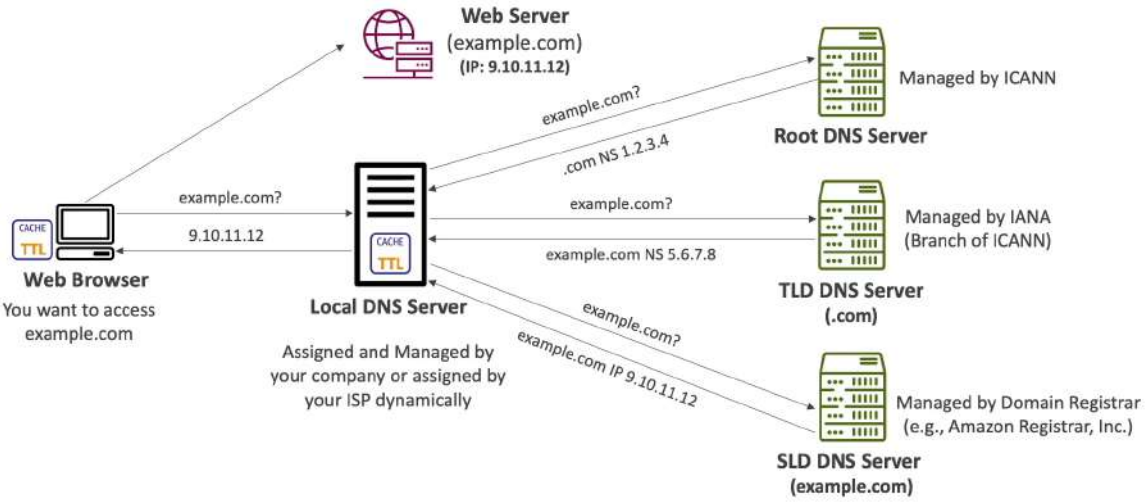
Route 53

▼ ☑ **DNS.**

DNS Terminologies

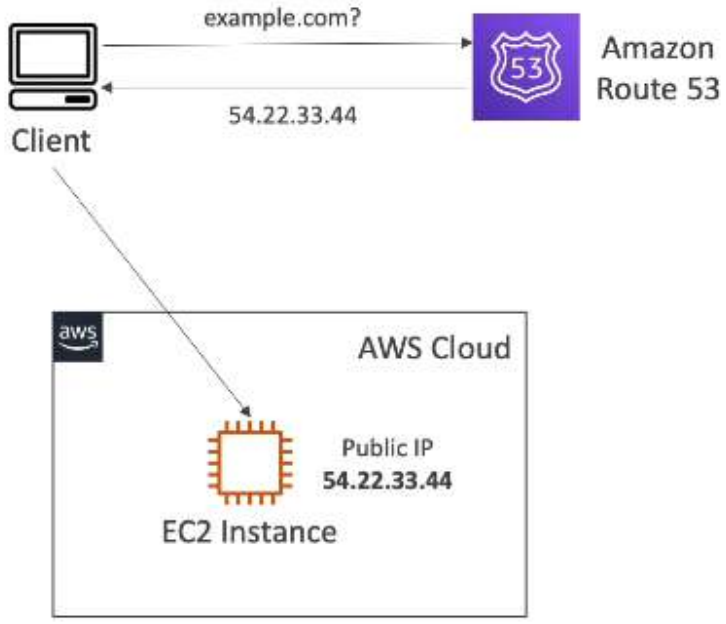
- Domain Registrar - Route 53, GoDaddy, etc...
- DNS Records - A, AAAA, CNAME, NS, etc...
- Zone File - contains DNS records.
- Name Server - resolves DNS queries (authoritative or non-autorithative).
- Top Level Domain (TLD) - .com, .us, .gov, .org ...
- Second Level Domain (SLD) - amazon.com, google.com.

How DNS Works



▼ ☑ **Route 53.**

Route 53 is a highly available, scalable, fully managed and authoritative (we can update the DNS records) DNS. It is also a **Domain Registrar** and we have ability to check the health of our resources.



Records define how we want to route traffic for a domain. Route 53 supports the following DNS record types: **A, AAA, CNAME, NS**, CAA, DS, MX, NAPTR, PTR, SOA, TXT, SPF, SRV.

Each record contains:

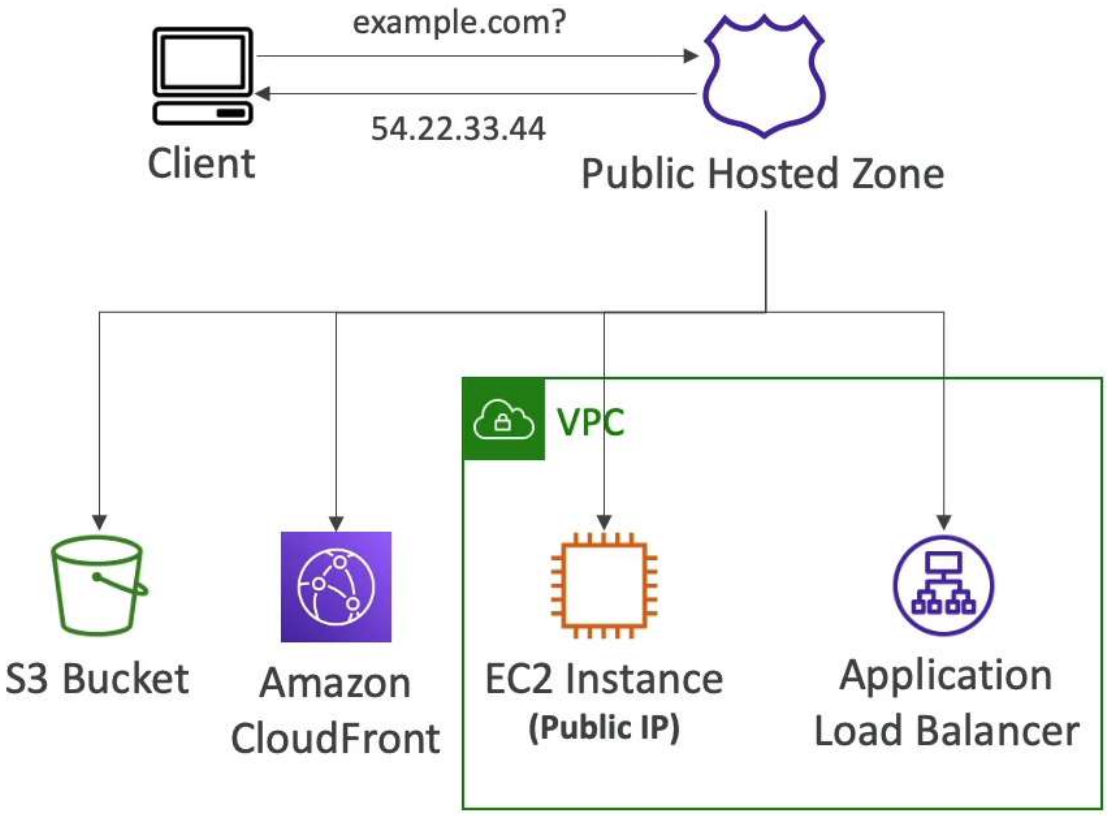
- **Domain/subdomain Name** (e.g. example.com)
- **Record Type** (e.g. A, AAAA)
- **Value** (e.g. 12.34.56.78)
- **Routing Policy** - how Route 53 responds to queries.
- **TTL (Time to Live)** - amount of time the record is cached at DNS Resolvers.

Record Types

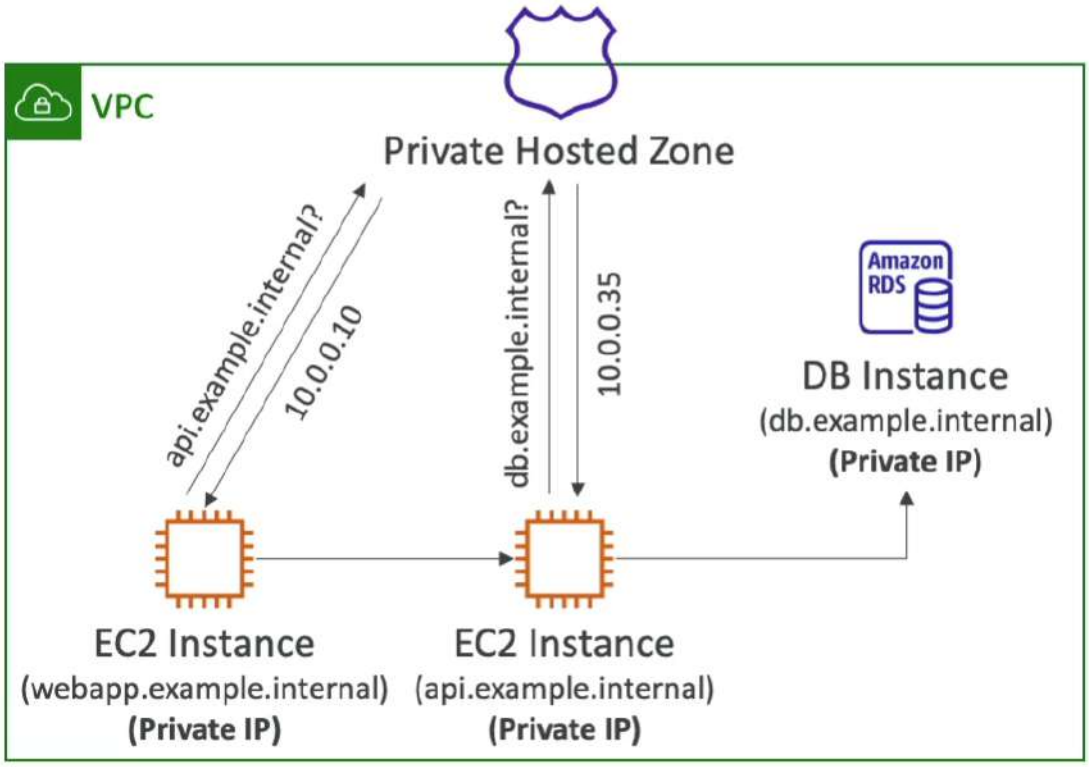
- A - maps a domain name to an IPv4 address.
- AAAA - maps a domain name to an IPv6 address.
- CNAME - maps hostname to another hostname. The target is a domain name which must have an A or AAAA record. We cant create a CNAME record for the top node of DNS namespace (Zone Apex).
- NS - specifies the authoritative DNS servers for the domain. For example, *example.com* might have NS records pointing to *ns1.example.com* and *ns2.example.com*.
- MX - specifies the mail server responsible for receiving emails for the domain.

Hosted Zones - a container for records that define how to route traffic to a domain and its subdomains. We pay $0.50 per month per hosted zone.

- **Public Hosted Zones** - contain records that specify how to route traffic on the internet.



- **Private Hosted Zones** - contain records that specify how we route traffic within one or more VPCs.



▼ **Routing Traffic to a Website Hosted in S3.**

The S3 bucket must have the same name as our domain or subdomain. Also we must have a registered domain name. We can use Route 53 as your domain registrar, or we can use a different registrar.

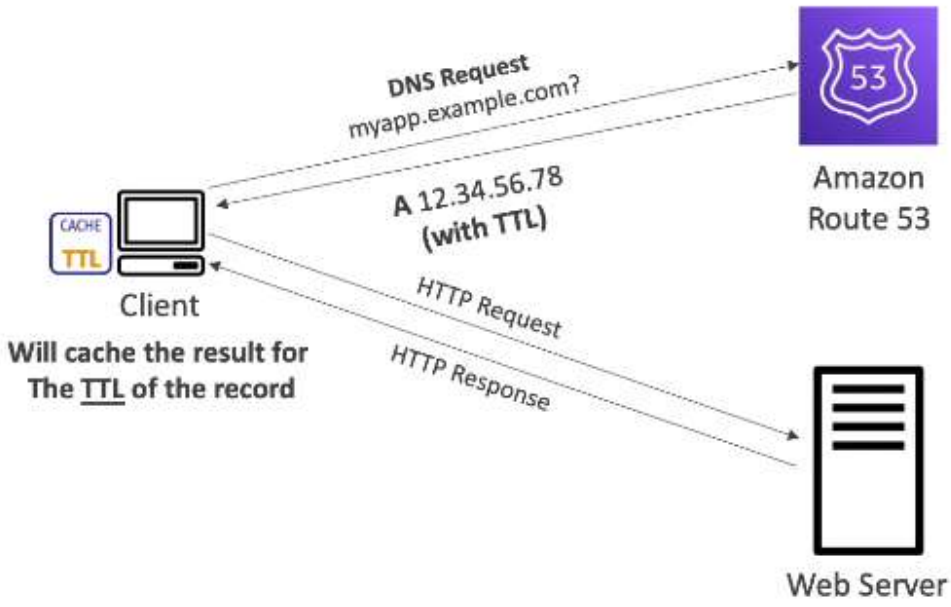
▼ ☑ **Route 53 TTL.**

TTL is a mechanism that specifies the duration that DNS resolvers should cache the information for a DNS record before querying the authoritative DNS server (Route 53) for updated information.

Short TTLs (e.g., 60 seconds) - for DNS records that may change frequently, such as records for load balancers or failover records. This allows changes to propagate quickly.

Long TTLs (e.g., 86400 seconds for 24 hours) - for stable records that rarely change. This reduces the number of DNS queries and can improve performance and reduce costs.

Except for Alias records, TTL is mandatory for each DNS record.



▼ ☑ **CNAME & Alias Records.**

AWS Resources expose an AWS hostname and if we want we can map that hostname to a domain we own.

CNAME - maps an alias domain name to a true or canonical domain name. Essentially, it redirects one domain name to another (app.mydomain.com ⇒ test.anything.com). Can be used only for non root domain.

Alias - specific to AWS Route 53 and provide a more powerful and flexible option compared to CNAME records. They can map domain names to AWS resources (app.mydomain.com ⇒ test.amazonaws.com). It works with root domain (apex domain) and non root domain. It is free of charge.

Alias record is always of type A/AAAA for AWS resources. It automatically recognizes changes in the resource's IP addresses. We cannot set the TTL when we have Alias record because Route 53 does it for us.

Alias Record Targets

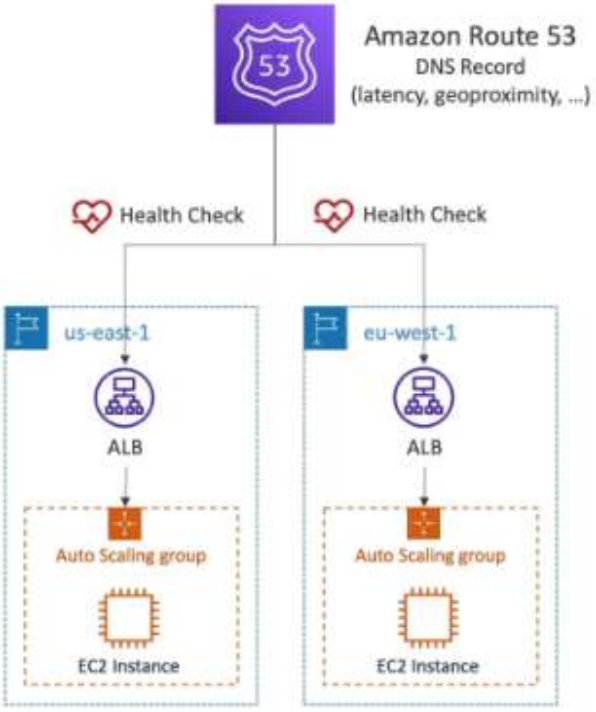
- Elastic Load Balancers
- CloudFront Distributions
- API Gateway
- Elastic Beanstalk environments
- S3 Websites
- VPC Interface Endpoints
- Global Accelerator accelerator
- Route 53 record in the same hosted zone

- You cannot set an ALIAS record for an EC2 DNS name

▼ ☑ **Route 53 Health Checks.**

HTTP Health Checks are only for public resources. With health checks we get automated DNS failover. Health checks can be integrated with CloudWatch

metrics.

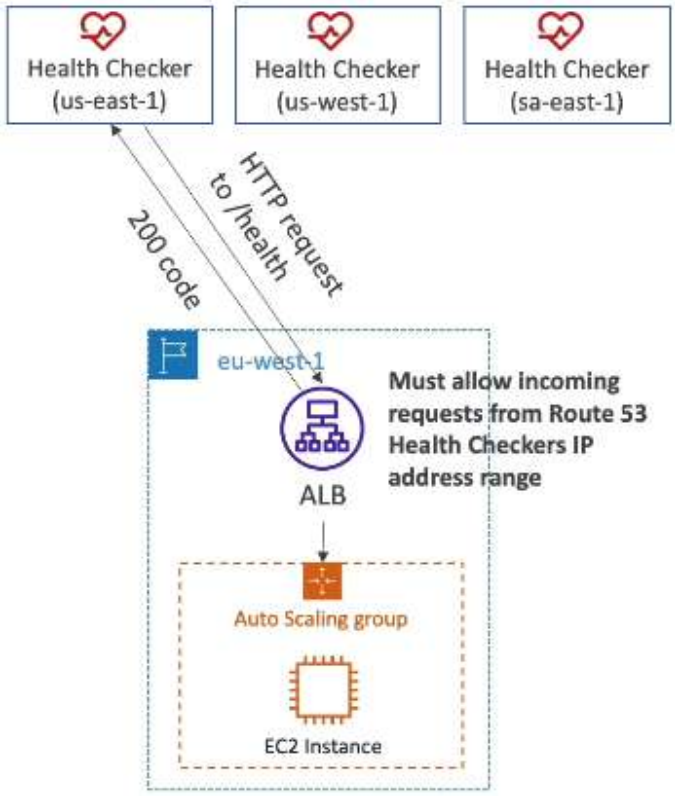


- Health Checks that monitor an enpoint

  About 15 global health checkers will check the endpoint health. If more than 18% of health checkers report the endpoint is healthy, Route 53 considers it Healthy.

  They pass only when the endpoint responds with the 2xx and 3xx status codes. Health checks can be setup to pass/fail based on the text in the first 5120 bytes of the response.

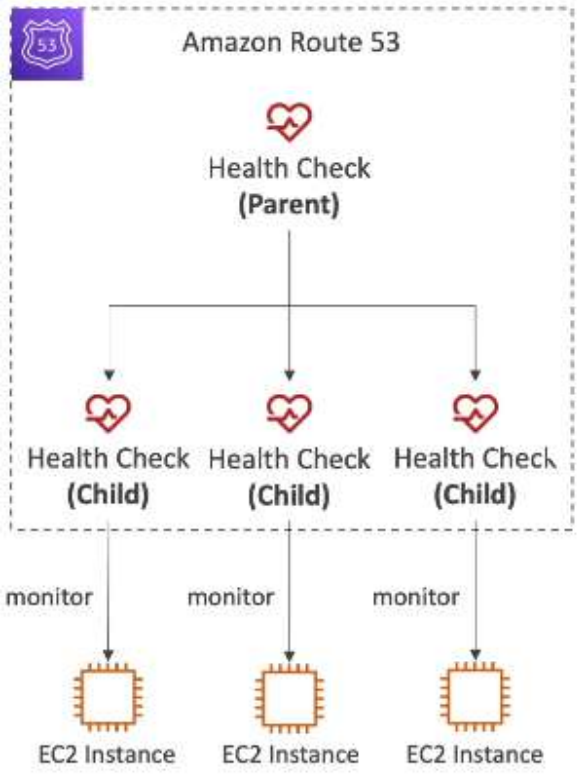
  To make it work we must configure our router/firewall to allow incoming requests from Route 53 Healtch Checkers.



- Calculated Health Checks

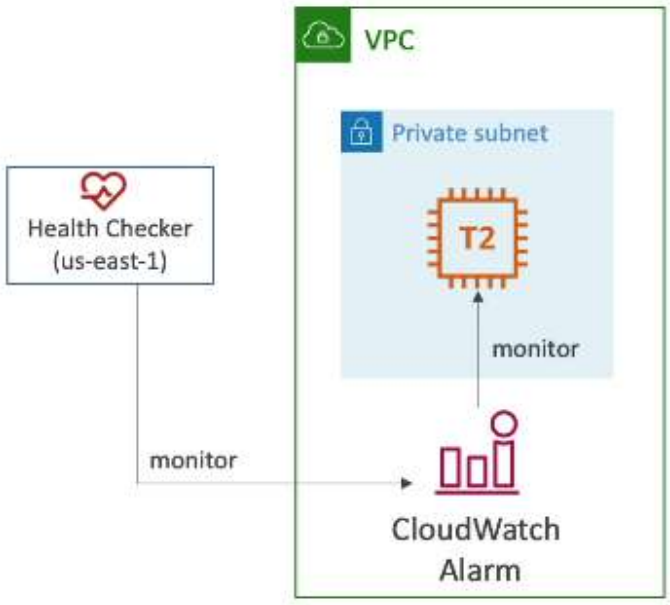
  It combines the reuslts of multiple health checks into a single health check. We can use OR, AND or NOT operators and can specify how many of the health checks need to pass to make the parent pass.

  It is used to perform maintenance to our website without causing all health checks to fail.



- Health Checks that monitor CloudWatch Alarms (**full control,** for private resources).

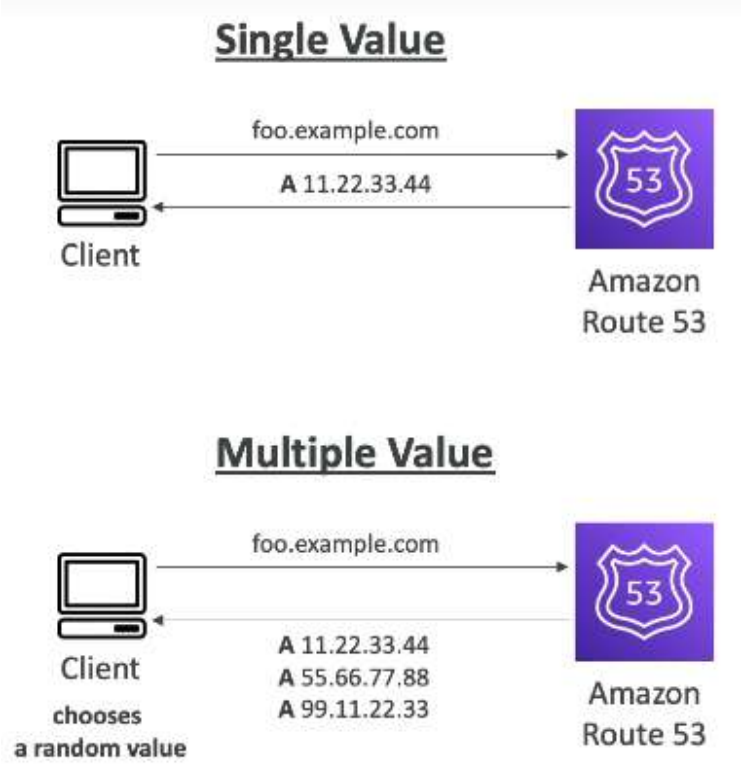
  Because health checkers are outside the VPC, they cant access private endpoints. We can create CloudWatch Metric and associate a CloudWatch Alarm, then create a health check that checks the alarm itself.



▼ ☑ **Routing Policy - Simple Routing.**

It routes traffic to a single resource. It can specify multiple values in the same record. If multiple values are returned, a random one is chosen by the client.

When Alias is enabled, we can specify only one AWS resource. It cannot be associated with Health Checks.



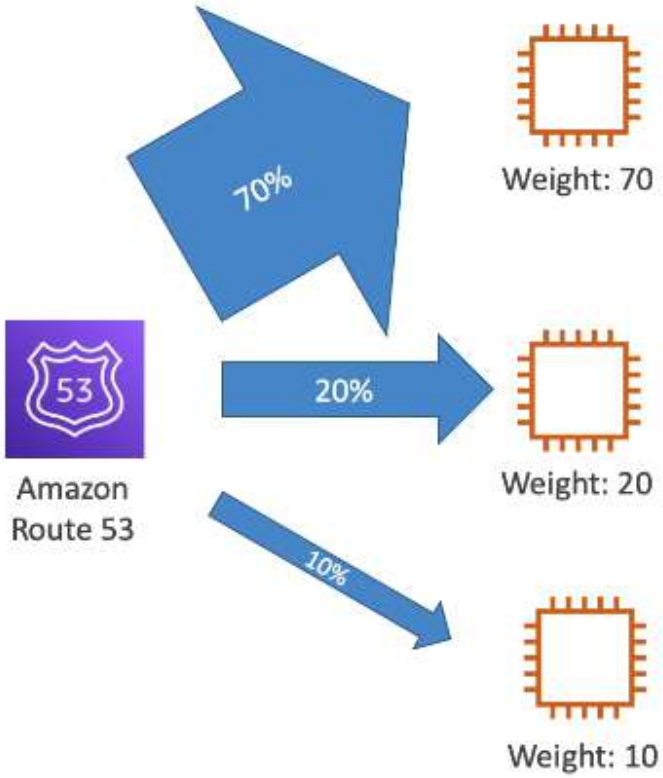
▼ ☑ **Routing Policy - Weighted Routing.**

It controls the percentage of the requests that go to each specific resource.

$$traffic = \frac{weight\ for\ a\ specific\ record}{sum\ of\ all\ the\ weights\ for\ all\ records}$$

DNS records must have the same name and type. This routing policy can be associated with Health Checks.

- **Assign a weight 0 to a record to stop sending traffic to a resource.**
- **If all records have weight of 0, then all records will be returned equally.**

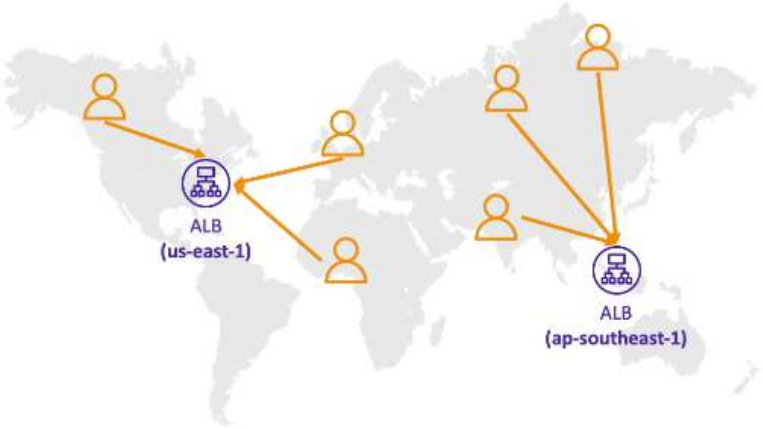


Use cases: load balancing between regions, testing new application versions ...

▼ ☑ **Routing Policy - Latency Routing.**

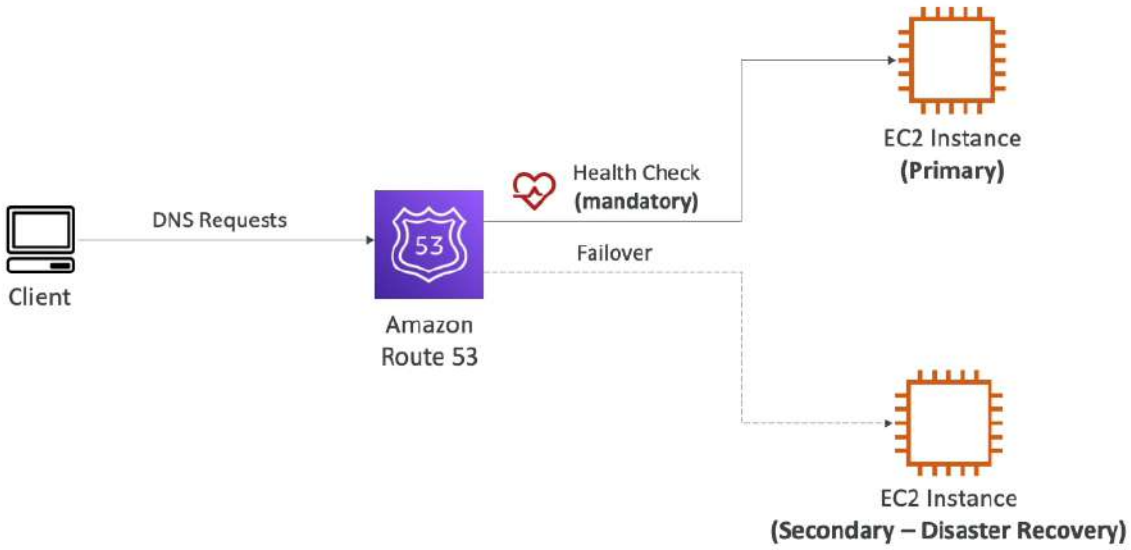
It redirects to the resource that has the least latency close to us. Latency is based on traffic between users and AWS regions.

Can be associated with Health Checks (has a failover capability).



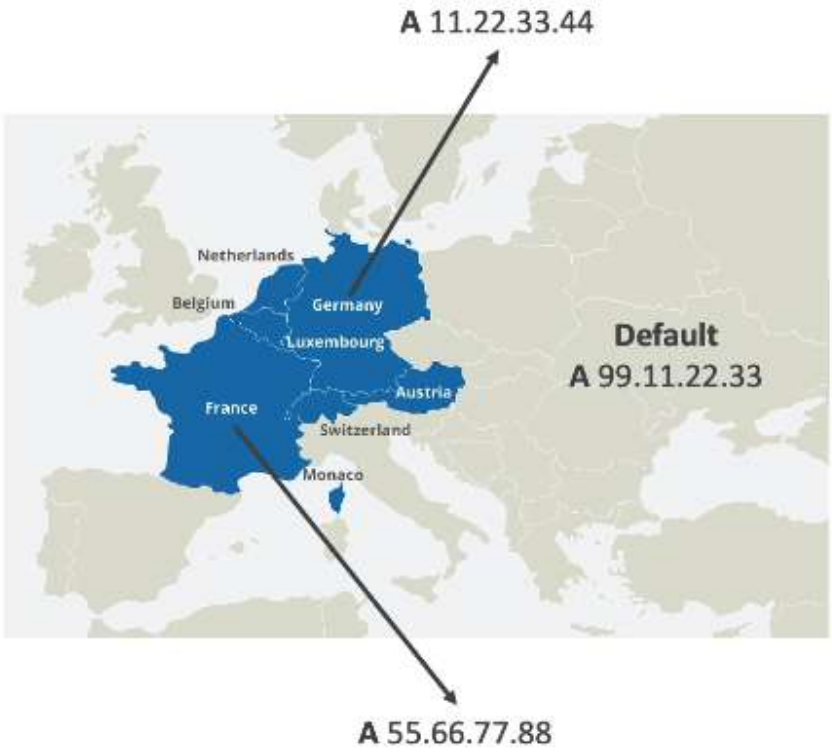
▼ ☑ **Routing Policy - Failover Routing.**

This routing policy is used to ensure high availability for applications by directing traffic to a backup site if the primary site becomes unavailable.



▼ ☑ **Routing Policy - Geolocation.**

It allows us to route traffic to different resources based on the geographic location of our users. This feature is useful for delivering content or services that are tailored to specific regions, optimizing performance, and improving the user experience (website localization, restrict content distributions).



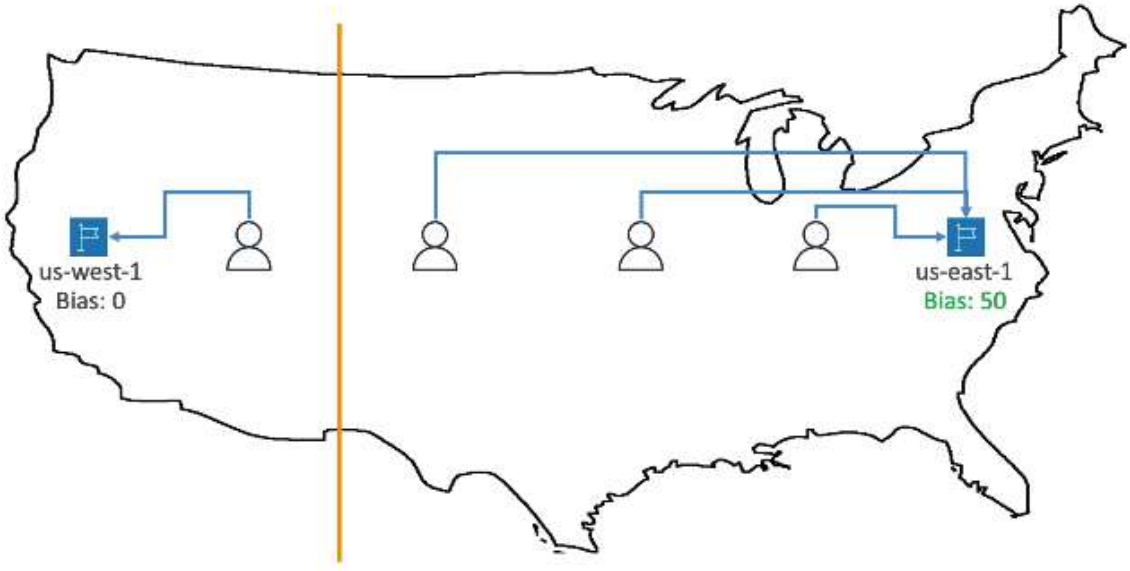
We should create a “Default” record (in case there is no match on location). Also, it can be associated with Health Checks.

▼ ☑ **Routing Policy - Geoproximity.**

It route traffic to our resources based on the geographic location of users and resources. We have ability to shift more traffic to resources based on the defined bias. Resources can be AWS resources (specified in aws region) and non-AWS resources (we need to specify latitude and longitude).

To change the size of the geographic region:

- 1≤ Bias ≤ 99 - more traffic to the resource.
- -1≤ Bias ≤ -99 - less traffic to the resource.



We must use Route 53 Traffic Flow to use this feature.

▼ ☑ **Routing Policy - IP-based Routing.**

It routes based on client's IP addresses. We need to provide a list of CIDRs for our clients and the corresponding endpoints.

It is used for performance optimization and reducing network costs.

▼ ☑ **Routing Policy - Multi-Value.**

It is used when routing traffic to multiple resources. Can be associated with Health Checks (return only values for healthy resources).

Name	Type	Value	TTL	Set ID	Health Check
www.example.com	A Record	192.0.2.2	60	Web1	A
www.example.com	A Record	198.51.100.2	60	Web2	B
www.example.com	A Record	203.0.113.2	60	Web3	C

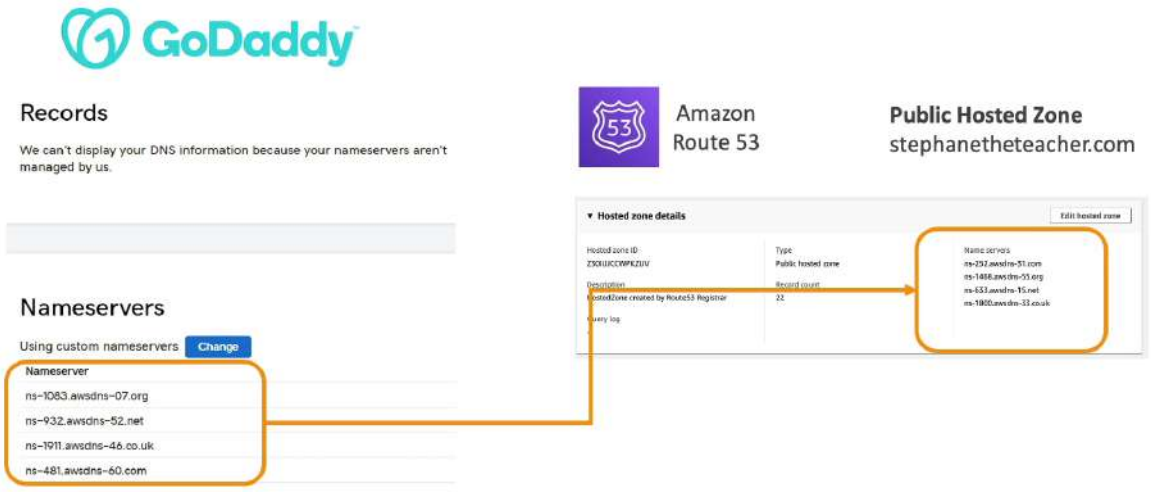
Multi-Value is not a substitute for having an ELB.

▼ ☑ **Third Party Domains & Route 53.**

We buy or register our domain name with a Domain Registrar typically by paying annual chargers. The Domain Registrar usually provides us with a DNS service to

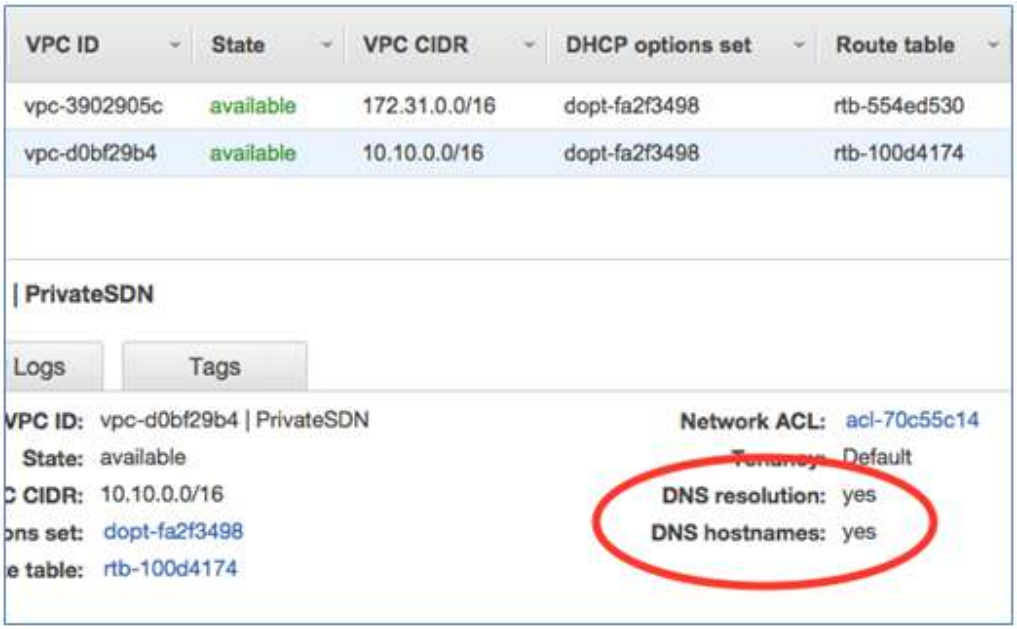
manage our DNS records, but we can use another DNS service to manage our DNS records.

If we buy a domain on a third party registrar, we can still use Route 53 as the DNS Service provider. First create a Hosted Zone in Route 53, then **update NS Records on third party website to use Route 53 Name Servers**.



▼ ☑ **VPC & DNS.**

When we launch an EC2 instance into a **default VPC**, AWS provides it with public and private DNS hostnames that correspond to the public IPv4 and private IPv4 addresses for the instance.



When we launch an instance into a **non-default VPC**, AWS provides the instance with a private DNS hostname only. New instances will only be provided with a public DNS hostname depending on these two DNS attributes: the **DNS resolution** and **DNS hostnames** that we have specified for our VPC and if our instance has a public IPv4 address.

Section II

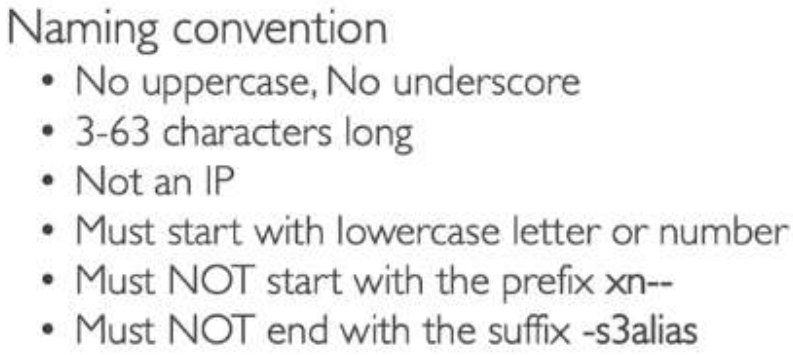
Amazon S3

▼ ☑ S3 Basics.

Amazon S3 is one of the main building blocks of AWS (advertised as "infinitely scaling" storage).

S3 use cases: backup and storage, disaster recovery, archive, hybrid cloud storage, application hosting, media hosting, data lakes, software delivery, static website ...

Amazon S3 allows people to store objects (files) in "**buckets**" (directories). Buckets must have a globally unique name (across all regions all accounts). Buckets are defined at the region level (S3 looks like a global service but buckets are created in a region).



Objects have a key. The key is FULL path (for example s3://my-bucket/folder1/folder2/file.txt). The key is composed of prefix+object name. There is no concept of “directories” within buckets, just keys with very long names that contain slashes.

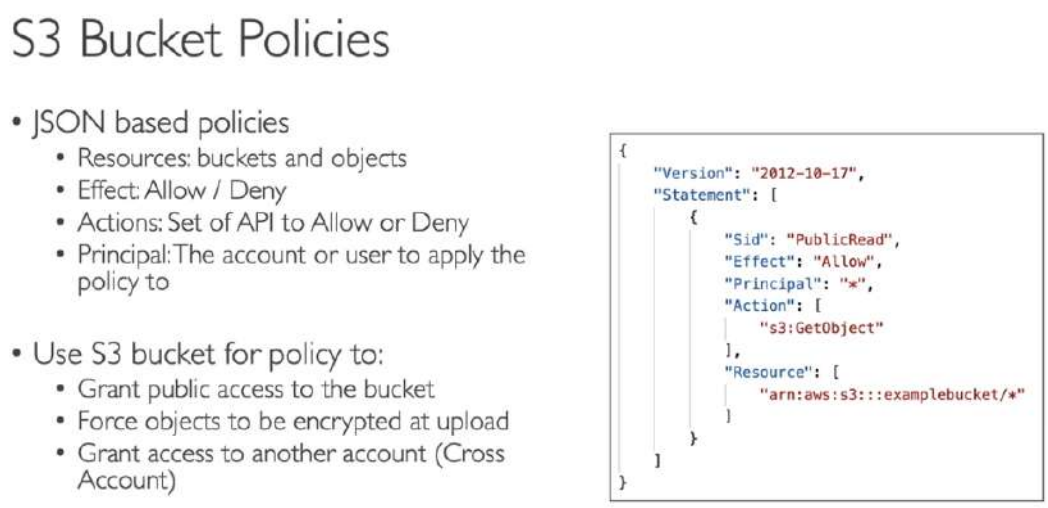
Object values are content of the body. Max object size is 5TB, if we upload more than 5GB, must use “multi-part upload”. Object can have also **metadata** (list of text key/value pairs), **tags** (unicode key/value pair, useful for security/lifecycle), **version ID**.

▼ ☑ **S3 Security.**

- **User-Based**

  IAM Policies - which API calls should be allowed for a specific user from IAM.

- **Resource-Based**
  - Bucket Policies - bucket wide rules from the S3 console (allows cross account).



- Object Access Control List (ACL) - finer grain (can be disable).
- Bucket Access Control List (ACL) - less common (can be disabled).

An IAM principal can access an S3 object if the user IAM permissions ALLOW it OR the resources policy ALLOWS it AND there is no explicit DENY.

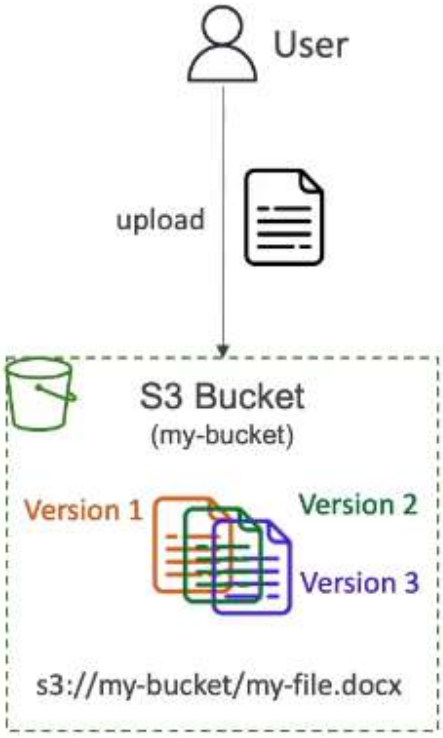
By default, an S3 object is owned by the AWS account that uploaded it even though the bucket is owned by another account. Object owner must explicitly grant the bucket owner access using a bucket policy to require external users to grant *bucket-owner-full-control* when uploading objects.

There are bucket settings for block public access. These settings were created to prevent company data leaks. If we know our bucket should never be public, we should leave these on (can be set at the account level).

> If we host static website and get 403 forbidden error, then we should check if bucket policy allows public reads.

▼ ☑ S3 Versioning.

Version for files can be enabled in Amazon S3. It is enabled at the bucket level. Same key overwrite will change the versions.



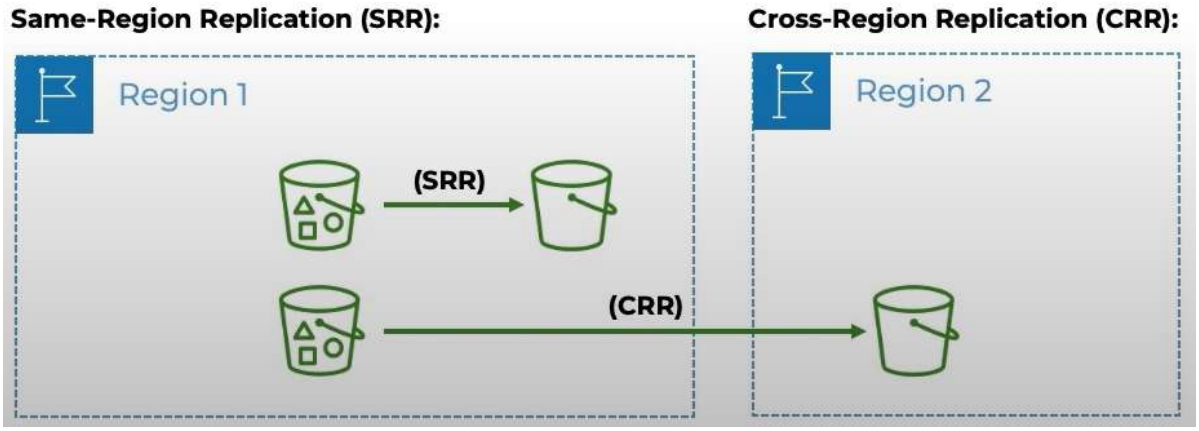
It's best practice to version our buckets. It protects us against unintended deletes (can restore a version) and we can roll back to previous version easily.

Any file that is not versioned prior to enabling versioning will have "null" version. Suspending versioning does not delete the previous versions.

▼ ☑ **S3 Replication (CRR & SRR).**

To perform replication we must enable versioning in source and destination buckets. Buckets can be in different AWS accounts. Copying is asynchronous and we must give proper IAM permissions to S3.

- **Cross-Region Replication (CRR)** - replicates objects between buckets in different AWS regions. It's useful for disaster recovery, latency reduction and compliance.
- **Same-Region Replication (SRR)** - replicates objects between buckets within the same AWS region. It's useful for log aggregation and live replication between production and test accounts.



▼ ☑ **S3 Storage Classes.**

- **S3 Standard - General Purpose**

  Used for frequently accessed data. Has low latency and high throughput. Sustain 2 concurent facility failures. It's used for big data analytics, mobile and gaming applications, content distribution ...

IA storage classes are used for data that is less frequently accessed, but requires rapid access when needed. They have lower cost than S3 Standard.

- **S3 Standard-IA**

  It's used for disaster recovery and backups. Has 99,9% availability.

- **S3 One Zone-IA**

  It's used for storing secondary backup copies of on-premise data, or data we can recreate. Has high durability in a single AZ. If AZ is destroyed data is lost.

S3 Glacier storage classes are low-cost object storage meant for archiving and backup. Pricing consist of price for storage + object retrieval cost.

- **S3 Glacier Instant Retrieval**

  Has millisecond retrieval time, great for data accessed once a quarter. Minimum storage duration is 90 days.

- **S3 Glacier Flexible Retrieval**

  - Expedited - 1 to 5 minutes retrieval time.

    Provisioned capacity ensures that our retrieval capacity for expedited retrievals is available when we need it. Each unit of capacity provides

that at least three expedited retrievals can be performed every five minutes and provides up to 150 MB/s of retrieval throughput.

  - Standard - 3 to 5 hours retrieval time.
  - Bulk - 5 to 12 hours retrieval time (free).

  Minimum storage duration is 90 days.

- **S3 Glacier Deep Archive**
  - Standard - 12 hours.
  - Bulk - 48 hours.

  Minimum storage duration is 180 days.

- **S3 Intelligent Tiering**

  Allows us to move objects between existing tiers based on usage pattern. We need to pay small montly monitoring and auto-tiering fee. There are no retrieval charges in S3 Intelligent-Tiering.

  - Frequent Access tier (automatic) - default tier.
  - IA tier (automatic) - objects not accessed for 30 days.
  - Archive Instant Access tier (automatic) - objects not accessed for 90 days.
  - Archive Access tier (optional) - configurable from 90 days to 700+ days.
  - Deep Archive Access tier (optional) - configurable from 180 days to 700+ days.

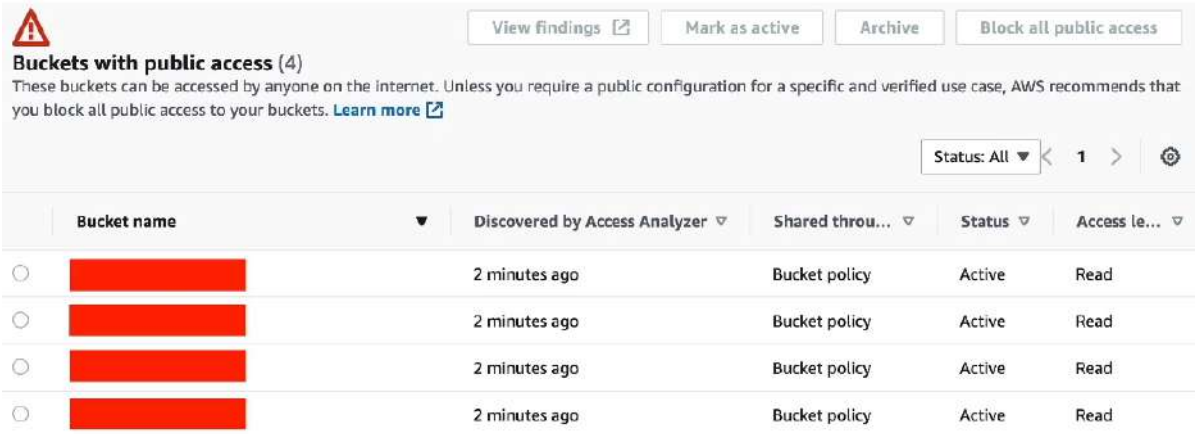
	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Durability	99.999999999% == (11 9's)						
Availability	99.99%	99.9%	99.9%	99.5%	99.9%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99%	99.9%	99.9%
Availability Zones	>= 3	>= 3	>= 3	1	>= 3	>= 3	>= 3
Min. Storage Duration Charge	None	None	30 Days	30 Days	90 Days	90 Days	180 Days
Min. Billable Object Size	None	None	128 KB	128 KB	128 KB	40 KB	40 KB
Retrieval Fee	None	None	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved

	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Storage Cost (per GB per month)	$0.023	$0.0025 - $0.023	%0.0125	$0.01	$0.004	$0.0036	$0.00099
Retrieval Cost (per 1000 request)	**GET:** $0.0004 **POST:** $0.005	**GET:** $0.0004 **POST:** $0.005	**GET:** $0.001 **POST:** $0.01	**GET:** $0.001 **POST:** $0.01	**GET:** $0.01 **POST:** $0.02	**GET:** $0.0004 **POST:** $0.03 **Expedited:** $10 **Standard:** $0.05 **Bulk:** free	**GET:** $0.0004 **POST:** $0.05 **Standard:** $0.10 **Bulk:** $0.025
Retrieval Time	Instantaneous					**Expedited** (1 – 5 mins) **Standard** (3 – 5 hours) Bulk (5 – 12 hours)	**Standard** (12 hours) **Bulk** (48 hours)
Monitoring Cost (pet 1000 objects)		$0.0025					

▼ ☑ **IAM Access Analyzer for S3.**

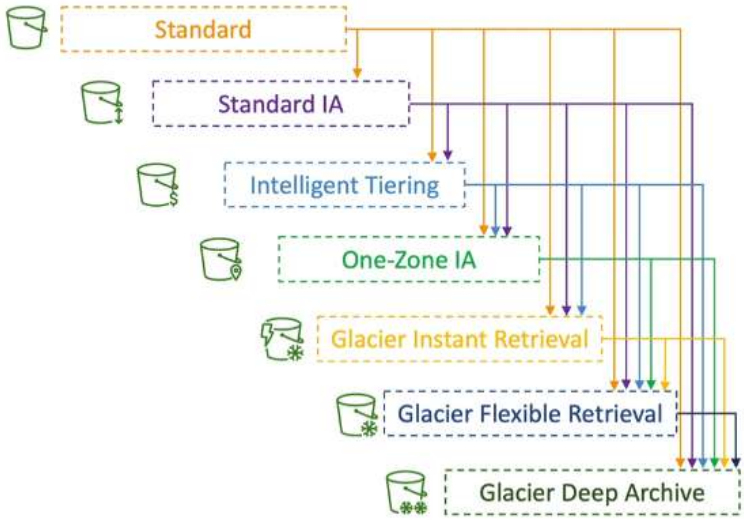
IAM Access Analyzer ensures that only intended people have access to our S3 buckets (for example: publicly accessible bucket or bucket shared with other AWS account).

It evaluates S3 Bucket Policies, S3 ACLs, S3 Access Point Policies.



▼ ☑ **S3 Lifecycle Rules.**

Lifecycle Rules in S3 are a set of actions that we can configure to manage the lifecycle of objects in our bucket. These rules enable us to automatically transition objects to different storage classes or delete them after a specified period. Lifecycle rules help us reduce costs and simplify storage management.



Transition Actions - used to configure objects to transition to another storage class.

Expiration Actions - used to configure objects to expire (delete) after some time.

- Can be used to delete old versions of files (if versioning is enabled).
- Can be used to delete incomplete Multi-Part uploads.

Rules can be created for a certain prefix or for certain objects tags.

▼ Lifecycle Rules Question- Scenario I

Our application on EC2 creates images thumbnails after profile photos are uploaded to S3. These thumbnails can be easily recreated and only need to be kept for 60 days. The source images should be able to be immediately retrieved for these 60 days, and afterwards, the user can wait up to 6 hours. How to design this?

- S3 source images can be on **Standard** with a lifecycle configuration to transition them to Glacier after 60 days.
- S3 thumbnails can be on **One-Zone IA** with a lifecycle configuration to expire them after 60 days.

▼ Lifecycle Rules Question- Scenario II

A rule in our company states that we should be able to recover our deleted S3 objects immediately for 30 days, although this may happen rarely. After

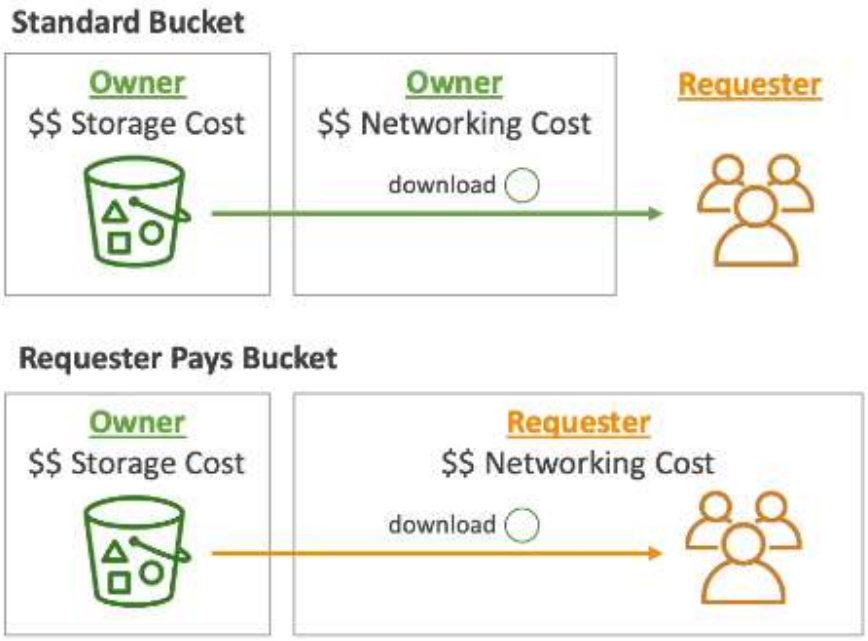
this time, and for up to 365 days, deleted objects should be recoverable within 48 hours.

- Enable S3 Versioning in order to have objects version.
- Transition the "noncurrent versions" of the object to Standard IA.
- Transition afterwards the "noncurrent versions" to Glacier Deep Archive.

S3 Analytics - helps us decide when to transition objects to the right storage class. Recommendations are only for Standard and Standard IA. Report is updated daily and we need to wait 24 to 48 hours to start seeing data analysis.

▼ ☑ **S3 Requester Pays.**

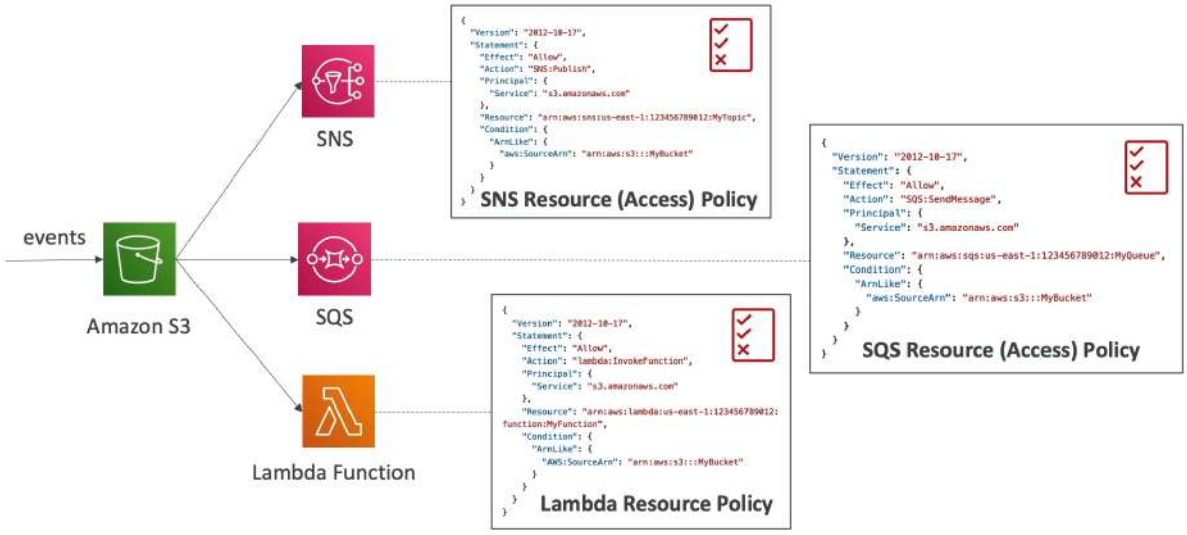
In general, buckets owners pay for all S3 storage and data transfer costs associated with their buckets. With Requester Pays buckets, instead of the bucket owner, the requester pays the cost of the request and the data download from the bucket.



The requester must be authenticated in AWS.

▼ ☑ **S3 Event Notifications.**

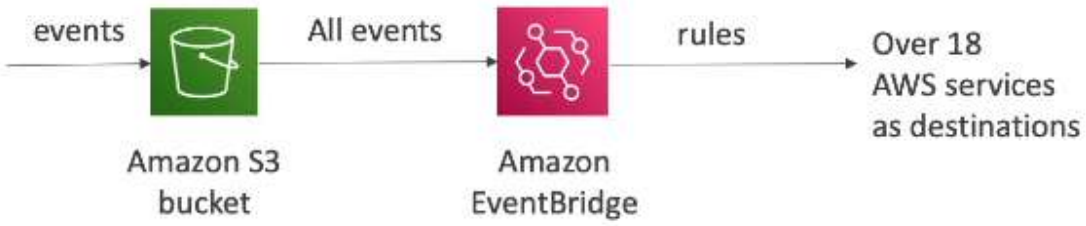
S3 Event Notifications allow us to receive notifications when certain events happen to objects within our S3 buckets. We can create as many S3 events as desired.



S3 event notifications typically deliver events in seconds but can sometimes take a minute or longer.

We can directly send S3 events to EventBridge, expanding the range of destinations and enabling more complex event handling and routing scenarios. SQS FIFO queues aren't supported as an S3 event notification destination.

- Advanced filtering options with JSON rules.
- Multiple destinations (Step Functions, Kinesis Streams)
- Archive and replay events.

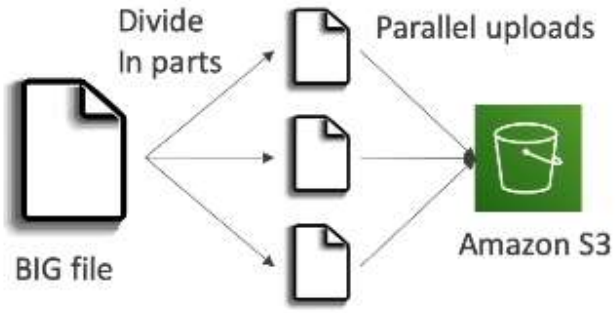


▼ ☑ **S3 Performance.**

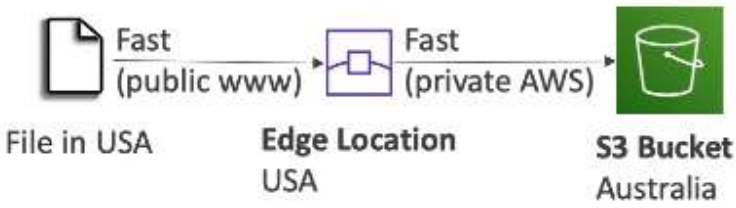
S3 automatically scales to high request rates, latency 100-200ms. Our application can achieve at least 3500 PUT/COPY/POST/DELETE or 5500 GET/HEAD requests per second per prefix in a bucket. If we spread reads across all four prefixes evenly, we can achieve 22000 requests per second for GET and HEAD.

Performance Boosts

- **Multi-Part Upload** - recommended for files < 100MB, must use for files >5GB. It can help parallelize uploads.



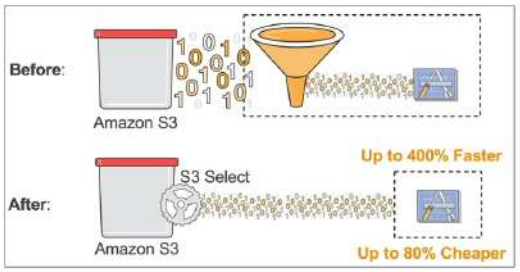
- **S3 Transfer Acceleration** - increase transfer speed by transferring file to an AWS Edge Location which will forward the data to the S3 bucket in the target region. It is compatible with multi-part upload.



- **S3 Byte-Range Fetches** - parallelizes GETs by requesting specific byte ranges. It has better resilience in case of failures. Can be used to speed up downloads and to retrieve only partial data.

▼ ☑ **S3 Select & Glacier Select.**

S3 Select & Glacier Select retrieves less data using SQL by performing server-side filtering. It can filter by rows and columns. We have less network transfer and less CPU cost on client-side. We can perform S3 Select to query only the necessary data inside the CSV files based on the bucket's name and the object's key.

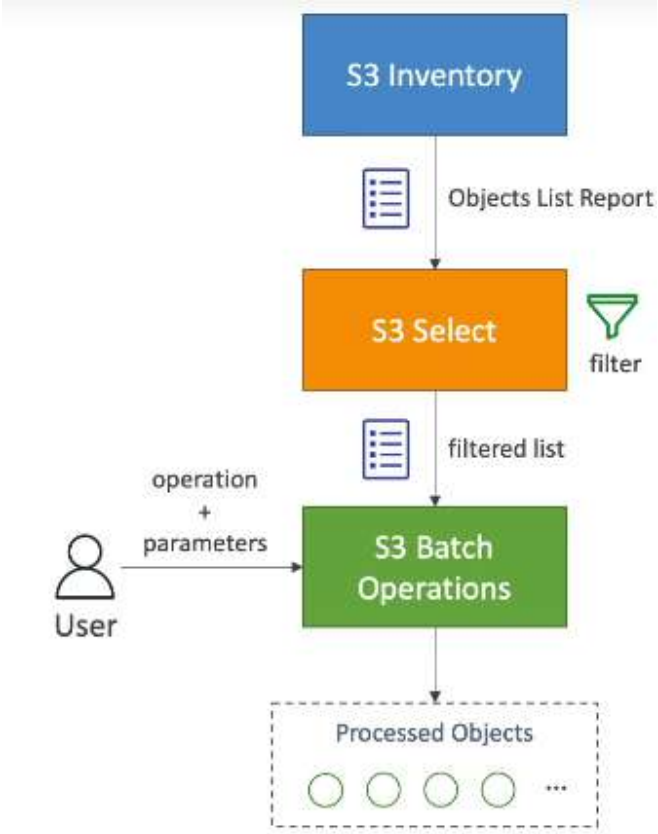


▼ ☑ **S3 Batch Operations.**

With **S3 Batch Operations** we can perform bulk operations on existing S3 objects with a single request. A job consists of a list of objects, the action to perform and optional parameters. S3 Batch Operations manages retries, tracks progress, sends completion notifications and generate reports.

Use cases:

- Modify object metadata & properties.
- Copy objects between S3 buckets.
- **Encrypt un-encrypted objects.**
- Modify ACLs
- Restore objects from S3 Glacier
- Invoke Lambda function to perform custom action on each object.



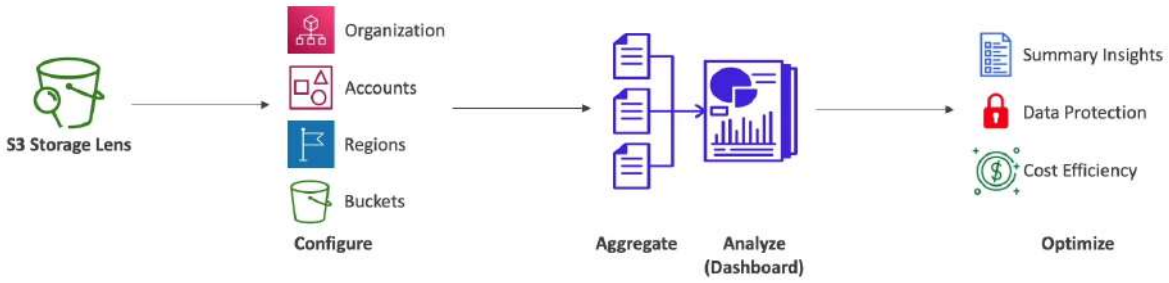
We can use S3 Inventory to get object list and use S3 Select to filter our objects.

▼ ☑ **S3 Storage Lens.**

With **Storage Lens** we can analzye and optimize storage across entire AWS Organization or AWS Accounts. We can discover anomalies, idenfity cost efficiencies and apply data protection best practices across entire AWS Organization.

It is possible to aggregate data for Organization, specific accounts, regions, buckets or prefixes.

Can be configured to export metrics daily to an S3 bucket (CSV, Parquet).



Default Dashboard - visualizes summarized insights and trends for both free and advanced metrics. It shows Multi-Region and Multi-Account data. It is preconfigured by S3 and cannot be deleted but can be disabled.

Storage Lens Metrics

- **Summary Metrics** - general insights about our S3 storage (StorageBytes, ObjectCount...). It is used for idenfitying the fastest-grownig (or not used) buckets and prefixes.
- **Cost-Optimization Metrics** - provide insights to manage and optimize our storage costs (NonCurrentVersionStorageBytes, IncompleteMultiPartUploadStorageBytes...). It is used for **identifying buckets with incomplete multipart** uploaded older than 7 days and which objects could be transitioned to lower cost storage class.
- **Data-Protection Metrics** - provide insights for data protection features. It is used for identifying buckets that arent following data-protection best practices.
- **Access-Management Metrics** - provide insights for S3 object ownership. It is used for identifying which object ownership settings our buckets use.
- **Event Metrics** - provide insights for S3 Event Notifications.
- **Performance Metrics** - provide insights for S3 Transfer Acceleration.
- **Activity Metrics** - provide insights about how our storage is requested.
- **Detailed Status Code Metrics** - provide insights for HTTP status codes.

Free Metrics - automatically available for all customers. Data is available for queries for 14 days.

Advanced Metrics and Recommendations - paid metrics and features (advanced metrics, CloudWatch publishing, prefix aggregation). Data is available for queries for 15 months.

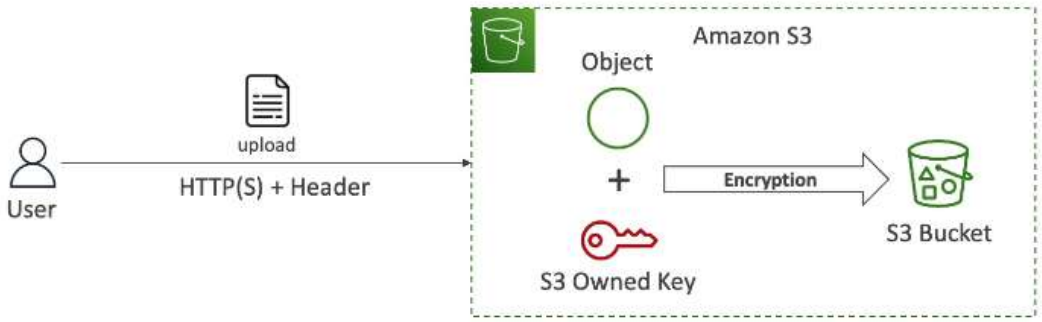
Amazon S3 Security

▼ ☑ **S3 Encryption.**

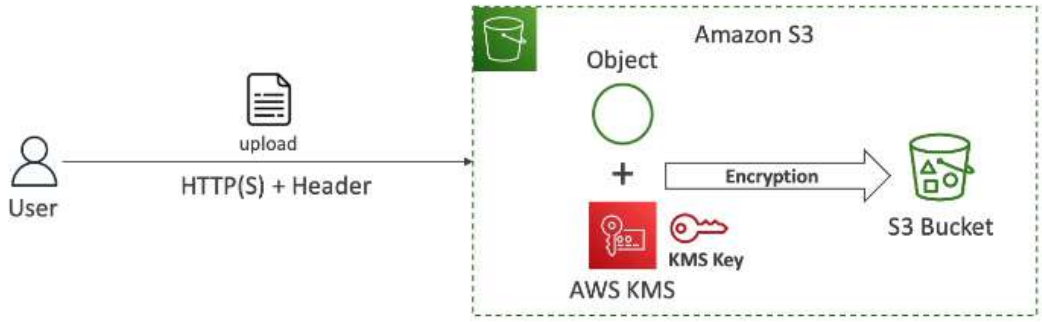
- **Server-Side Encryption (SSE)**

- **SSE with S3-Managed Keys (SSE-S3)** - it is enabled by default and encrypts S3 objects using keys handled, managed and owned by AWS.

  We must set header *"x-amz-server-side-encryption":"AES256"*.



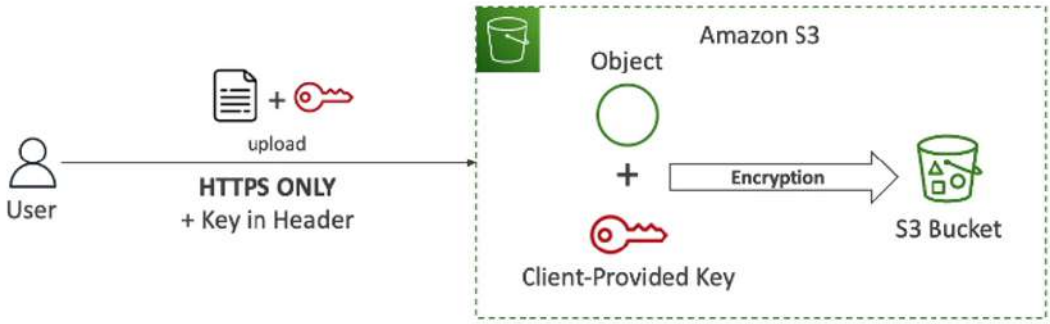
- **SSE with KMS Keys stored in AWS KMS (SSE-KMS)** - leverages KMS to manage encryption keys.
- We must set header *"x-amz-server-side-encryption":"aws:kms"*.



If we use SSE-KMS we may be impacted by the KMS limits. When we upload, it calls the GenerateDataKey KMS API and when we download, it calls the Decrypt KMS API.

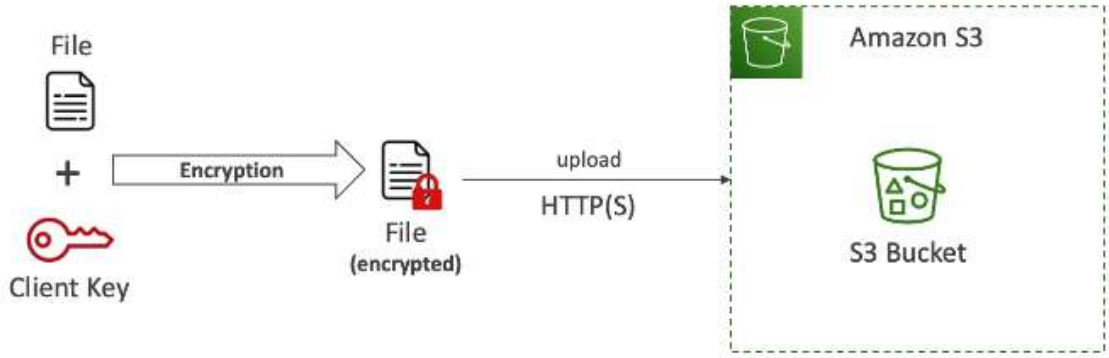
- **SSE with Customer-Provided Keys (SSE-C)** - when we want to manage our own encryption keys.

  We must use HTTPS for this type of encryption. Encryption key must be provided in HTTP headers, for every HTTP request made.



- **Client-Side Encryption**

  This type of encryption leverages client libraries such as S3 Client-Side Encryption Library. Clients must encrypt (decrypt) data themselves before sending (retrieving) to (from) S3. Customer fully manages the keys and encryption cycle.



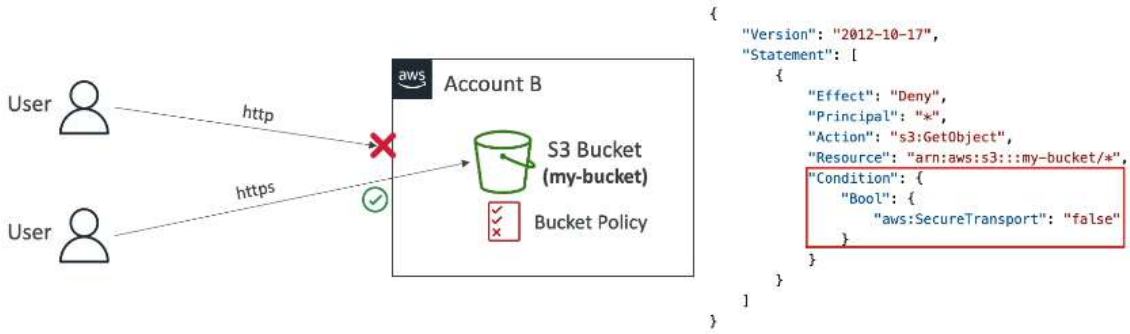
- **Encryption in Transit**

  S3 exposes two endpoints:

  - HTTP Endpoint (not encrypted)
  - HTTPS Endpoint (encryption in flight)

  HTTPS is mandatory for SSE-C.

  Force Encryption in Transit



SSE-S3 encryption is automatically applied to new objects stored in S3 buckets. Optionally, we can force encryption using a bucket policy and refuse any API call to PUT an S3 object without encryption headers (SSE-KMS or SSE-C).

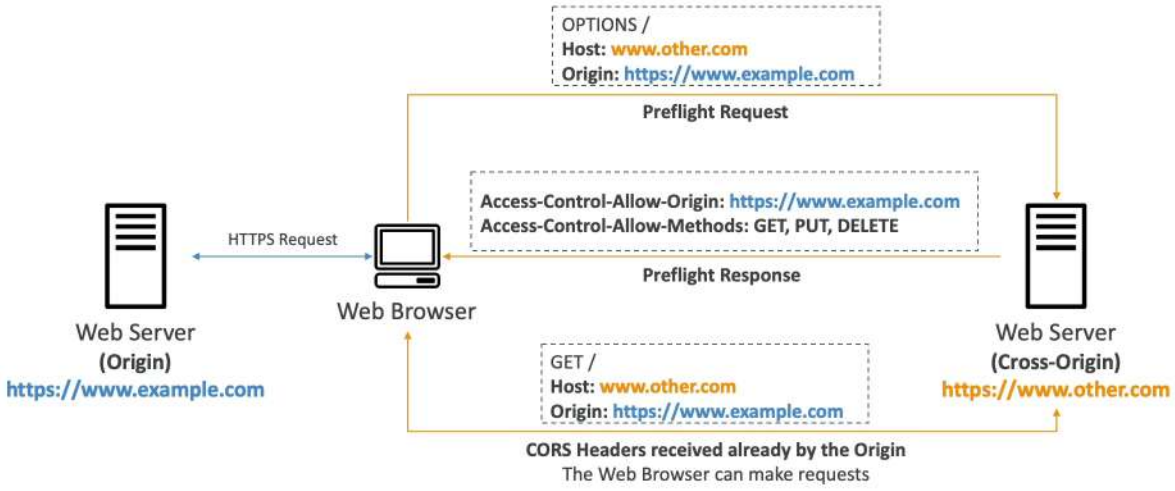
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "s3:PutObject",
            "Principal": "*",
            "Resource": "arn:aws:s3:::my-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "s3:PutObject",
            "Principal": "*",
            "Resource": "arn:aws:s3:::my-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "true"
                }
            }
        }
    ]
}
```

Bucket Policies are evaluated before “Default Encryption”.

▼ ☑ **S3 CORS.**

CORS - Web Browser based mechanism to allow requests to other origins while visiting the main origin. The request wont be fulfilled unless the other origin allows for the requests, using CORS Headers (Access-Control-Allow-Origin).



If a client makes a cross-origin request on our S3 bucket, we need to enable the correct CORS headers. We can allow for a specific origin or * for all origins.

▼ ☑ S3 MFA Delete.

MFA Delete is a feature designed to add an extra layer of security to protect our S3 bucket's objects from accidental or malicious deletion. It requires users to provide two or more forms of verification before they can perform deletion.

To use MFA Delete, Versioning must be enabled on the bucket and only the bucket owner (root account) can enable/disable MFA Delete.

▼ ☑ S3 Access Logs.

S3 Access Logs are used for audit purpose, we may want to log all access to S3 buckets. Any request made to S3, from any account, authorized or denied will be logged into another S3 bucket. Target logging bucket must be in the same AWS region.

> We should not set our logging bucket to be the monitored bucket. It will create a logging loop and our bucket will grow exponentially.

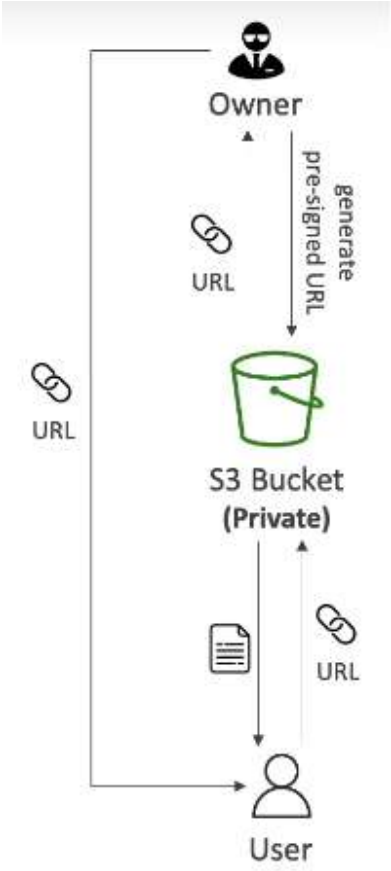
▼ ☑ S3 Pre-Signed URLs.

We can generate pre-signed URLs using the S3 Console, AWS CLI or SKD. Users given a pre-signed URL inherit the permissions of the user that generated the URL for GET and PUT methods.

Use cases:

- Allow only logged-in users to download a premium video from our S3 bucket.
- Allow an ever-changing list of users to download files by generating URLs dynamically.

- Allow temporarily a user to upload a file to a precise location in our S3 bucket.



Pre-signed URLs **allow users to interact directly with the cloud storage service rather than routing through our application server**. This can reduce latency.

▼ ☑ **S3 Glacier Vault Lock & Object Lock.**

S3 Glacier Vault Lock lets us adopt a WORM (Write Once Read Many) model. We need to create a Vault Lock Policy. It is helpful for compliance and data retention.

S3 Object Lock lets us adopt a WORM model and block an object version deletion for a specified amount of time.

- **Retention mode - Compliance**

  Object versions cant be overwritten or deleted by any user, including the root users. Objects retention modes cant be changed and retention periods cant be shortened.

- **Retention mode - Governance**

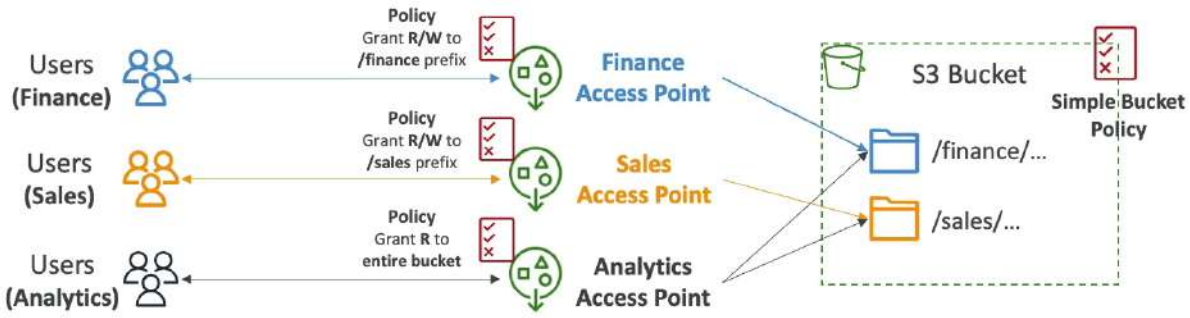
  Most users cant overwrite or delete an object version or alter its lock settings. Some users have special permissions to change the retention or delete the objects.

Retention Period - protect the object for a fixed period, it can be extended.

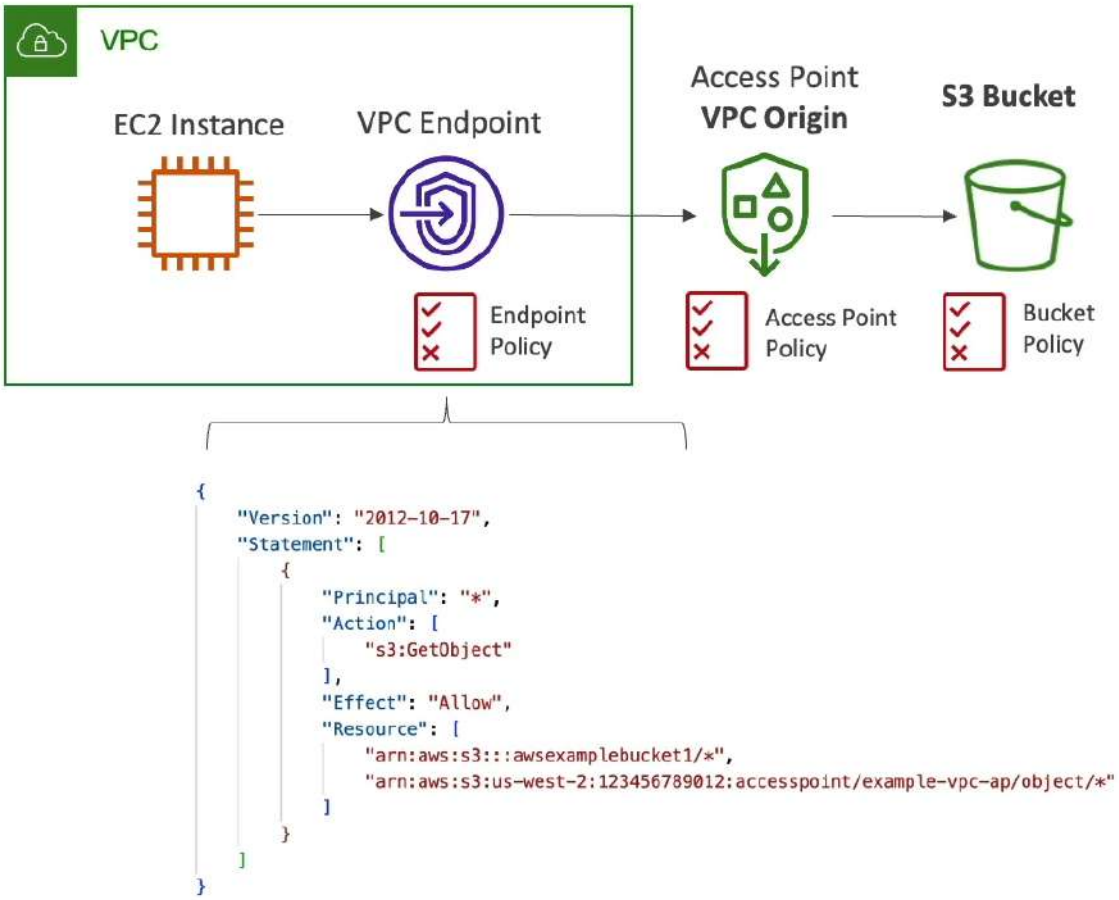
Legal Hold - protect the object idefinitely, independent from retention period.

▼ ☑ **S3 Access Points.**

Access Points simplify security management for S3 Buckets. Each Access Point has its own DNS name (Internet Origin or VPC Origin) and access point policy (similar to bucket policy).



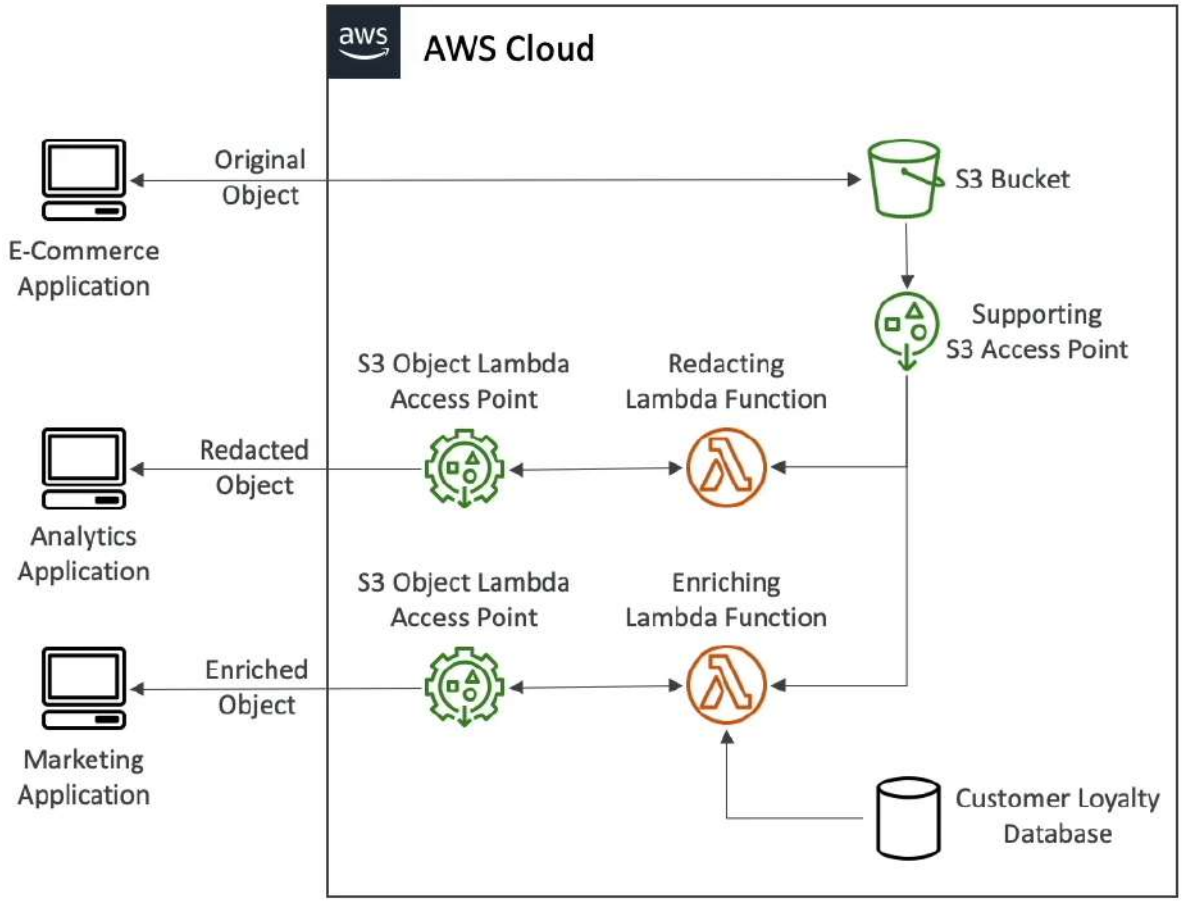
We **can define the access point to be accessible only from within the VPC**. To do that we must create a VPC Endpoint to access the Access Point and the VPC Endpoint must allow access to the target bucket and Access Point.



S3 Multi-Region Access Points - provide a global endpoint that applications can use to fulfill requests from S3 buckets located in multiple AWS Regions. We can use Multi-Region Access Points to build multi-Region applications with the same simple architecture used in a single Region, and then run those applications anywhere in the world. They provide built-in network resilience with acceleration of internet-based requests to Amazon S3. Application requests made to a Multi-Region Access Point global endpoint use AWS Global Accelerator.

▼ ☑ **S3 Object Lambda.**

We can use AWS Lambda Functions to change the object before it is retrieved by the caller application. Only one S3 bucket is needed, on top of which we create S3 Access Point and S3 Object Lambda Access Points.



Use cases:

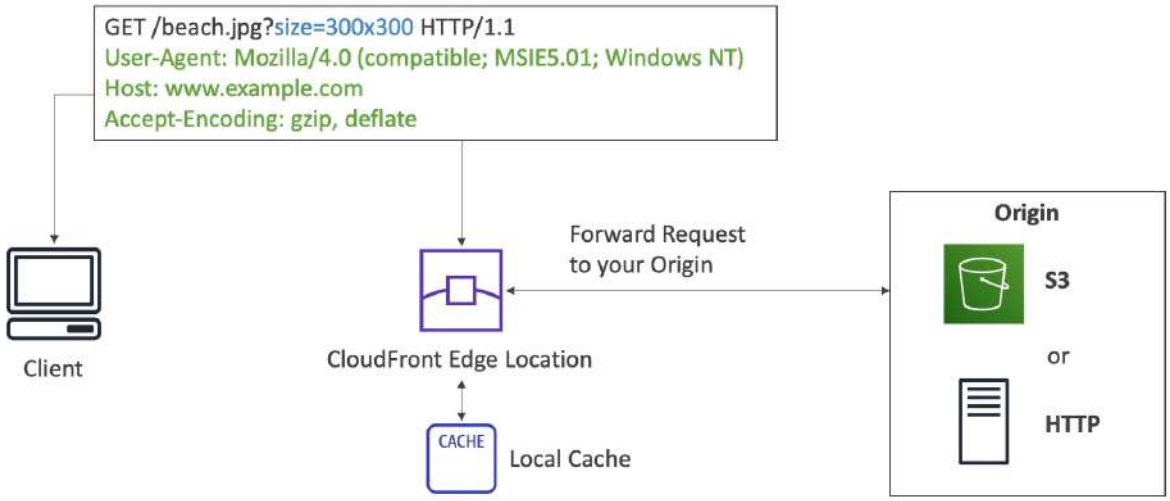
- Redacting PII information from analytics or non-productiuon environments.
- Converting across data formats, such as converting XML to JSON.
- Resizing and watermarking images on the fly using caller-specific details, such as the user who requested the object.

CloudFront, Global Accelerator & Wavelength

▼ ☑ **AWS CloudFront.**

AWS CloudFront is a Content Delivery Network (CDN) service. It delivers our content to end-users with low latency and high transfer speeds.. It improves read performance, content is cached at the edge location. When a user requests content, CloudFront serves it from the nearest edge location.

CloudFront integrates with AWS Shield for DDoS protection, AWS Web Application Firewall (WAF) to protect against common web exploits, and SSL/TLS encryption to secure data transfer.



CloudFront Access Control

CloudFront signed URLs and signed cookies allow us to control who can access our content.

Signed URLs use cases:

- RTMP distribution (signed cookies aren't supported for RTMP distributions).
- Restrict access to individual files (e.g. an installation download for your application).
- Users are using a client that doesn't support cookies.

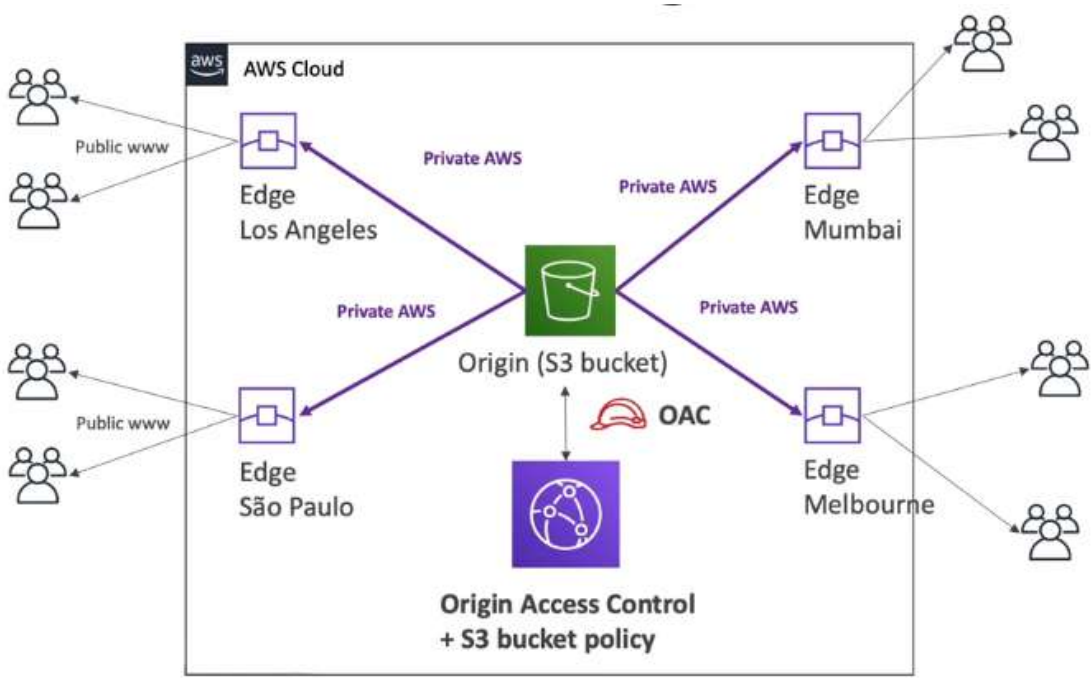
Signed Cookies use cases:

- Provide access to multiple restricted files (e.g. all of the files for a video in HLS format or all of the files in the subscribers' area of a website).
- We don't want to change our current URLs.

CloudFront Origins

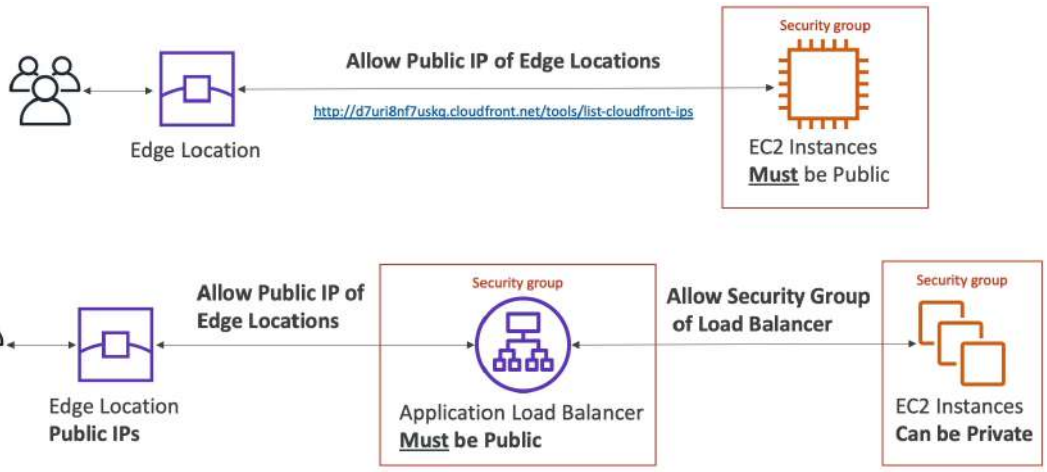
- S3 bucket

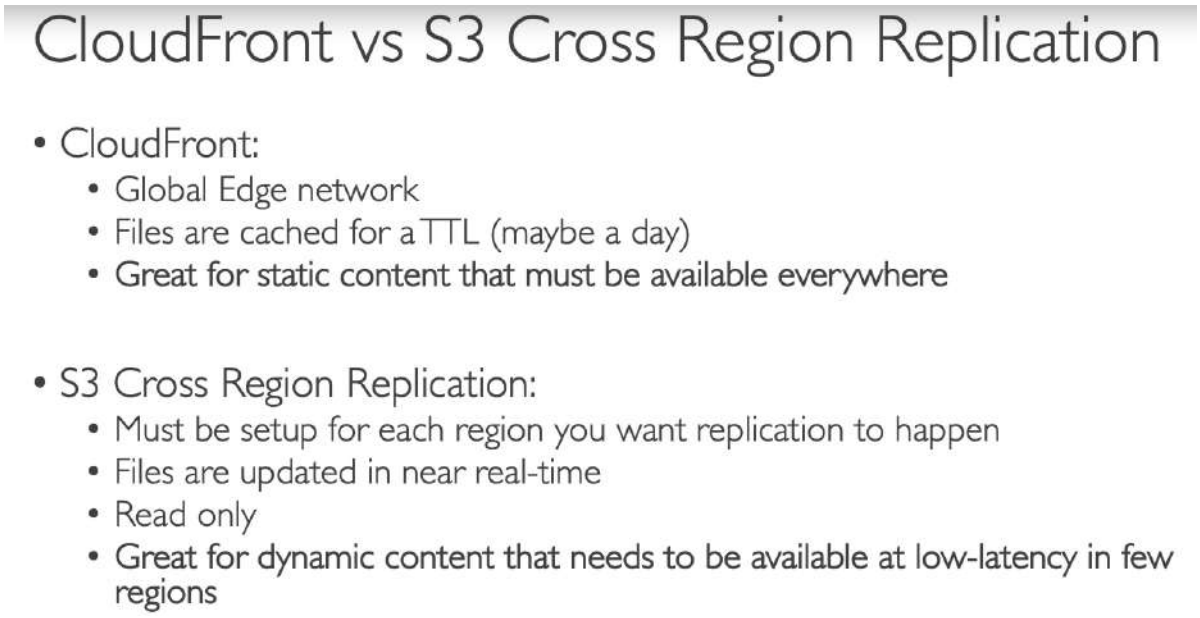
Used for distributing files and caching them at the edge location. Enhanced security with CloudFront **Origin Access Control (OAC)**. It can be used as an ingress (to upload files to S3).



- Custom Origin (HTTP)

  It can be **ALB, EC2 instance or S3 website** (must enable the bucket as a static S3 website).





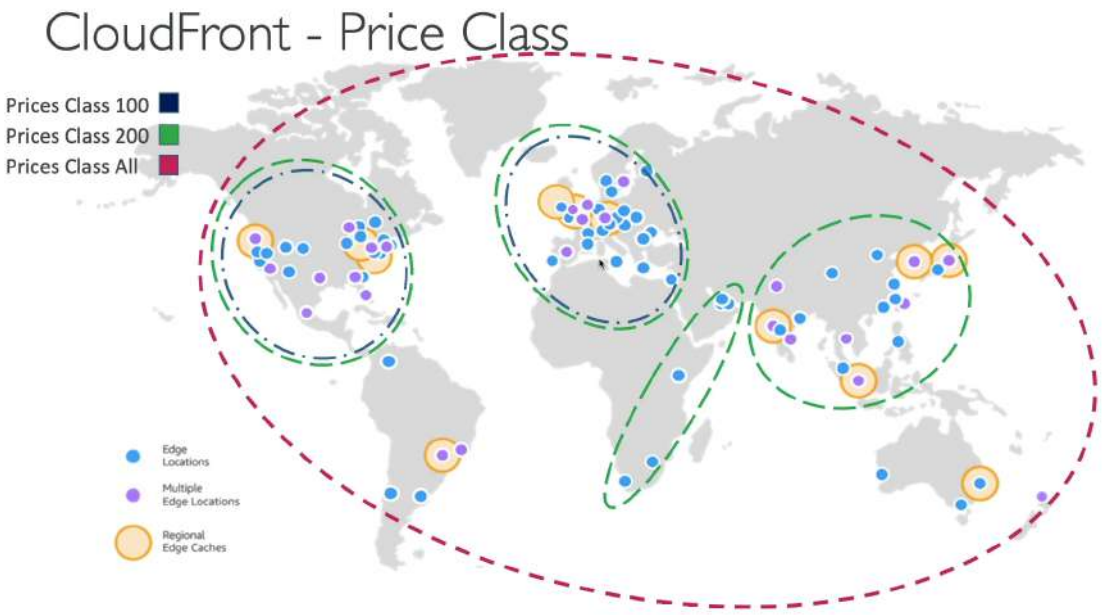
Origin Access Identity (OAI) is a special CloudFront user that we can create to give CloudFront permission to access our content stored in an S3 bucket.

▼ ☑ **CloudFront Geo Restriction.**

We can restrict who can access our distribution based on geographic location. The country is determined using a third party Geo-IP database.

- **Allowlist** - allow our users to access our content only if they are in one of the countries on a list of approved countries.
- **Blocklist** - prevent our users from accessing our content if they are in one of the countries on a list of banned countries.

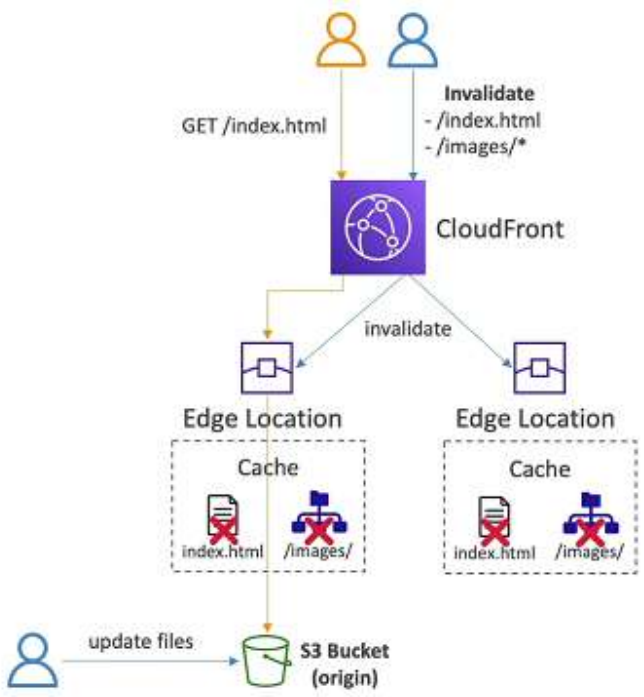
▼ ☑ **CloudFront Price Classes.**



▼ ☑ **CloudFront Cache Invalidation.**

The Cache-Control and Expires headers control how long objects stay in the cache. The **Cache-Control max-age** directive lets us specify how long (in seconds) we want an object to remain in the cache before CloudFront gets the object again from the origin server. The minimum expiration time CloudFront supports is 0 seconds for web distributions and 3600 seconds for RTMP distributions.

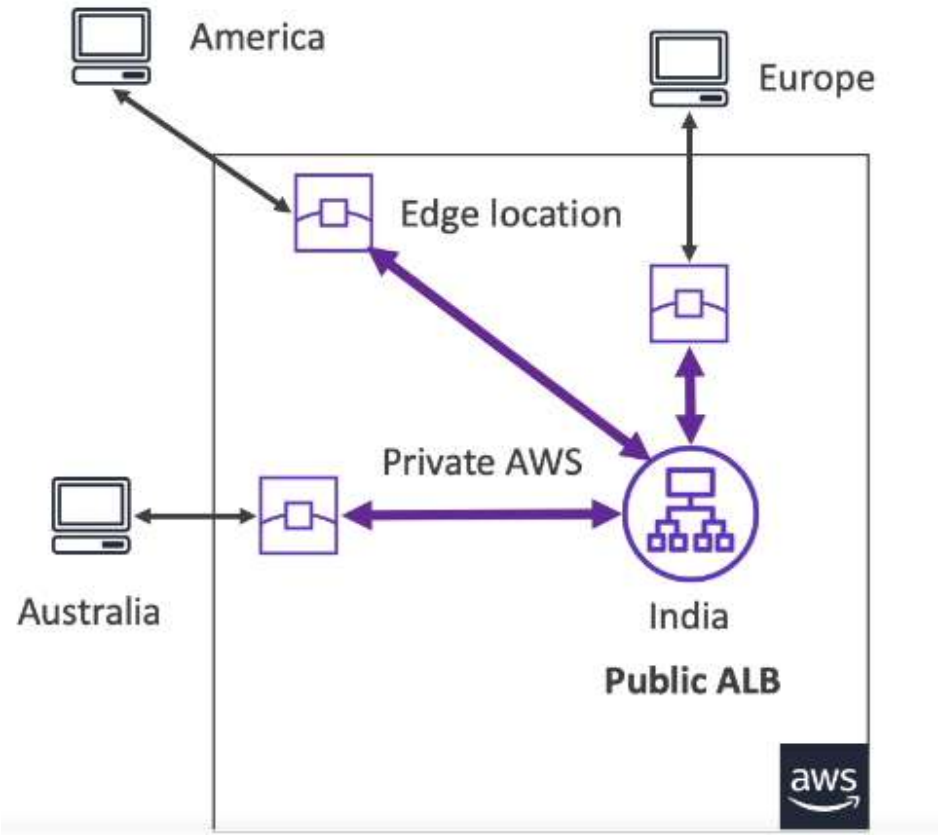
In case we update the back-end origin, CloudFront doesnt know about it and will only get the refreshed content after the TTL has expired. We can force an entire or partial cache refresh by performing a CloudFront Invalidation. We can invalidate all files or special path.



▼ ☑ **AWS Global Accelerator.**

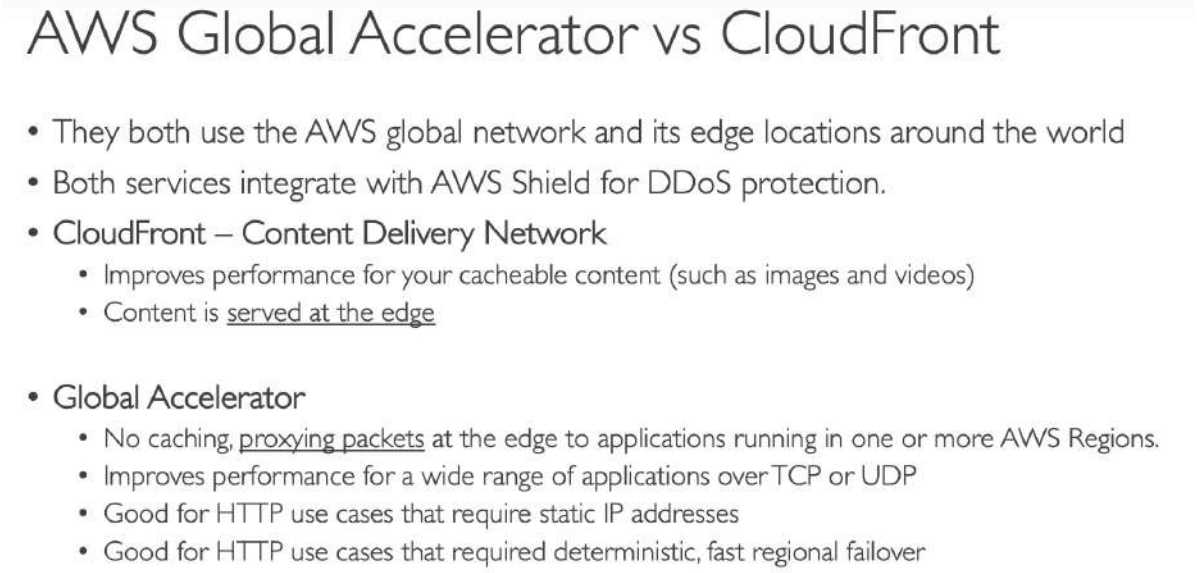
AWS Global Accelerator - improves global application availability and performance using the AWS global network. It leverages the AWS internal network to optimize the route to our application (60% improvement).

It provides two static IP addresses that serve as a fixed entry point to our application. These IP addresses are anycast from AWS edge locations, ensuring user traffic is directed to the closest AWS region.



It works with Elastic IP, EC2 instances, ALB, NLB. We can use AWS Shield for DDoS protection.

Global Accelerator performs a health check of our applications. It helps make our application global (failover less than 1 minute for unhealthy) and it's great for disaster recovery.



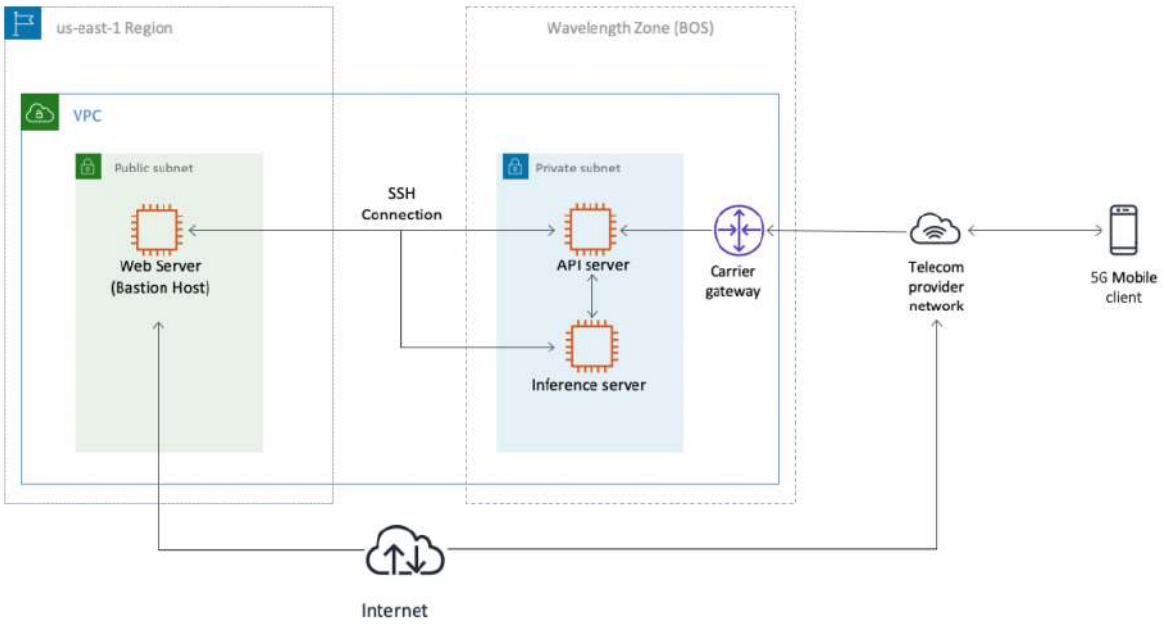
▼ ☑ **AWS Wavelength.**

AWS Wavelength is a service that brings AWS compute and storage services to the edge of the 5G network. It enables developers to build ultra-low-latency applications by deploying them closer to end-users, specifically within the infrastructure of telecom providers.

Wavelength Zones are specific infrastructure deployments that host AWS compute and storage services within the 5G networks. These zones are deployed in partnership with telecom providers like Verizon, Vodafone, KDDI, and SK Telecom, among others. We can extend our VPC into Wavelength Zones.

Carrier Gateway - networking component used specifically in Wavelength Zones to enable communication between our VPC and devices connected to 5G networks. It provides a route for traffic between your VPC and the telecom provider's 5G network, allowing your Wavelength-hosted applications to communicate with devices using carrier-assigned IP addresses.

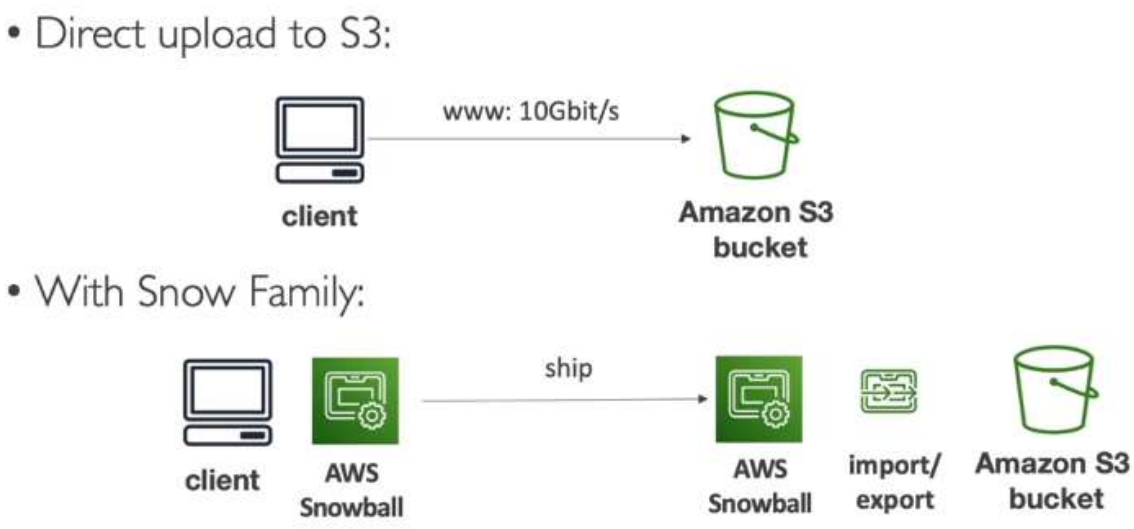
Communication between applications and 5G devices can remain private within the telecom provider's network without ever needing to cross the public internet.



AWS Storage Extras

▼ ☑ **AWS Snow Family.**

AWS Snow Family are highly-secure, portable devices which collect and process data at the edge and migrate data into and out of AWS. If it takes more than a week to transfer over the network, we should use Snowball devices.



- **Snowball Edge**

  This a physical data transport solution. We can move TBs or PBs of data in or out of AWS. It's alternative to moving data over the network and paying network fees. Instead we pay per data transfer job. Typical use cases are large data cloud migrations and disaster recovery.

  Snowball Edge proivides **block storage** and Amazon **S3** compatibile object storage.

  - **Snowball Edge Storage Optimized**

    80TB of HDD or 210TB NVMe capacity for block volume and S3 compatible object storage.

  - **Snowball Edge Compute Optimized**

    42TB of HDD or 28TB NVMe capacity for block volume and S3 compatible object storage.

- **Snowcone**

  It's a small, portable, rugged & secure device. Withstands hars enviroments. Typically used for edge computing, storage and data transfer. When Snowball does not fit we should use Snowcone (we must provide our own battery and cables). It can be sent back to AWS offline or connect it to internet and use AWS DataSync to send data.

  - **Snowcone** - 8TB of HDD Storage

    - **Snowcone SSD** - 14TB od SSD Storage

- **Snowmobile**

  It's a truck which can transfer exabytes of data. Each Snowmobile has 100PB capacity. If we transfer more thatn 10PB it is better than Snowball.

	Snowcone & Snowcone SSD	Snowball Edge Storage Optimized	Snowmobile
Storage Capacity	8 TB HDD 14 TB SSD	80 TB - 210 TB	< 100 PB
Migration Size	Up to 24 TB, online and offline	Up to petabytes, offline	Up to exabytes, offline
DataSync agent	Pre-installed		

Snowbal Edge & Snowcone can be also used in edge computing (processing data while it's being created on an edge location). Use cases of edge computing: preprocessing data, ML, transcoding media streams ...

- **Snowcone & Snowcone SSD** ⇒ 2CPUs, 4GB of RAM, can use USB-C.
- **Snowball Edge - Compute Optimized** ⇒ 104CPUs, 416GB of RAM, optional GPU, 28TB NVMe or 42TB HDD.
- **Snowball Edge - Storage Optimized** ⇒ 40CPUs, 80GB of RAM, 80TB storage OR 104CPUs, 416GB of RAM, 210TB NVMe storage

All can run EC2 Instances & AWS Lambda functions (using AWS IoT Greengrass). There are long-term deployment options - 1 and 3 years discounted pricing. Snowcone is better for IoT.

▼ **Solution Architecture - Snowball into Glacier**

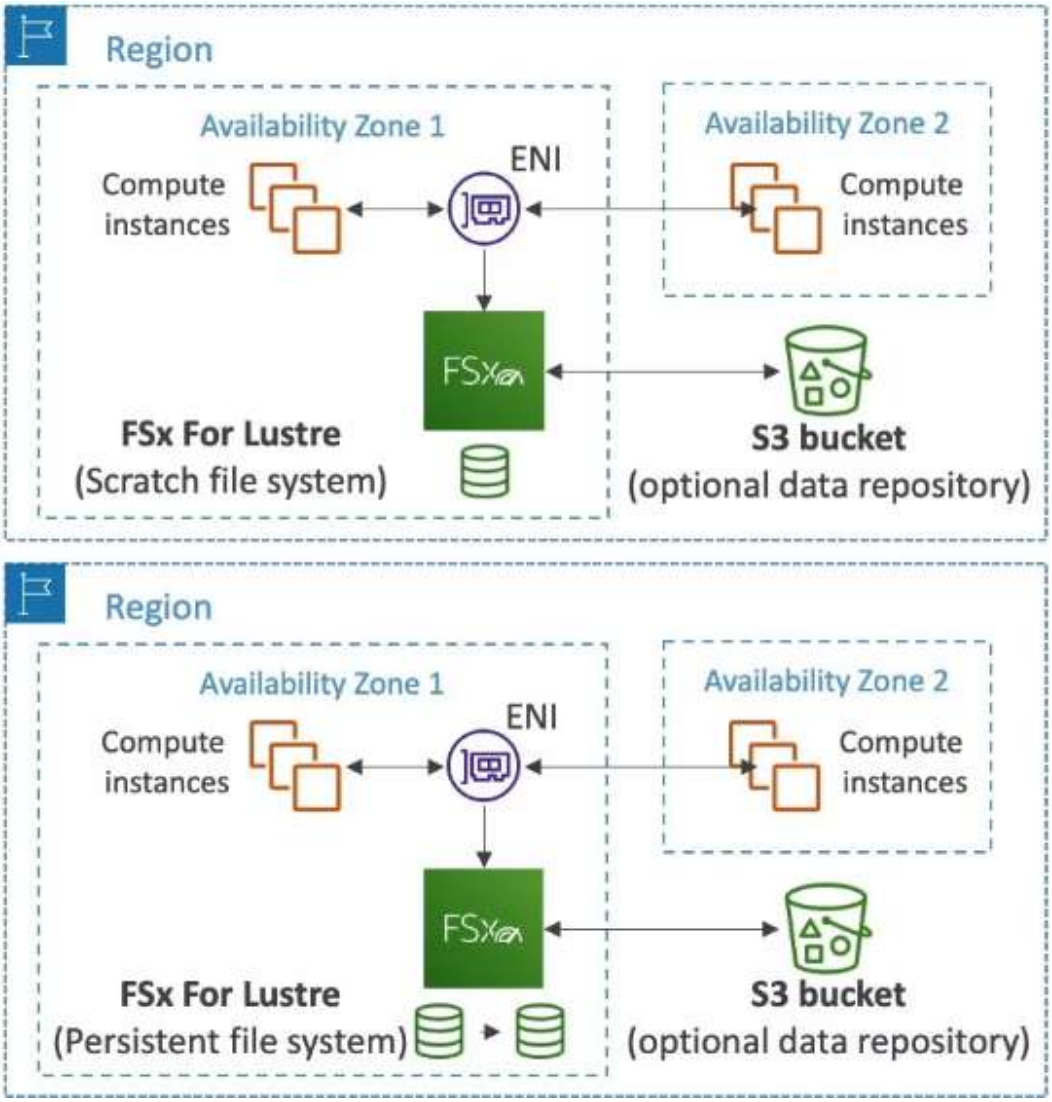
Snowball cannot import to Glacier directly. We must use S3 first in combination with an S3 lifecycle policy.

▼ ☑ **Amazon FSx.**

Amazon FSx is a fully managed service that lets us launch third party high-performance file systems on AWS.

FSx System Deployment Options

- **Scratch File System** - temporary storage. Data is not replicated (doesn't persist if file server fails). It has high burst and it is used for short-term processing.
- **Persistent File System** - long-term storage. Data is replicated within same AZ. It replaces failed files within minutes. Used for long-term processing and sensitive data.

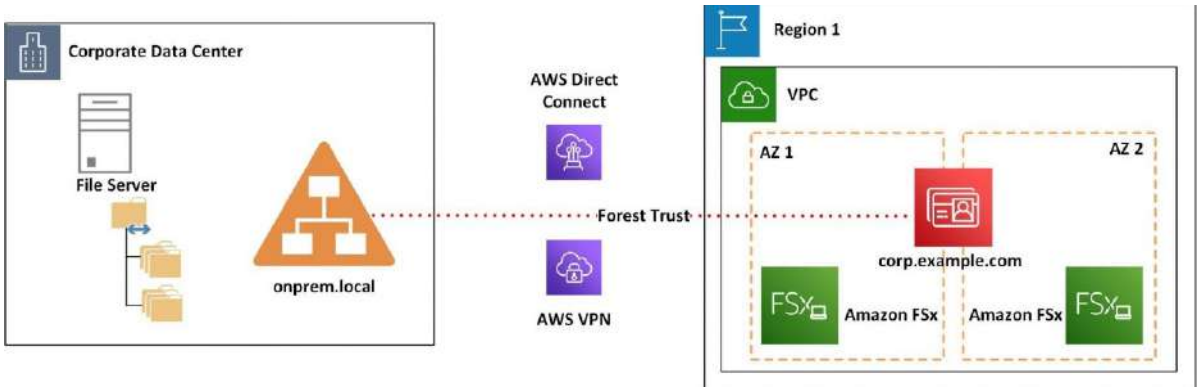


▼ ☑ **FSx - Windows File Server.**

FSx for Windows File Server - supports SMB and NTFS protocols. It can be mounted on Linux EC2 instances. We can integrate it with Microsoft AD, ACLs and user quotas. It supports Microsoft's Distributed File System (DFS) Namespaces.

We can choose SSD or HDD for storage options. FSx for Windows can be accessed from our on-premises infrastructure (VPN or DX). Can be configured to be Multi-AZ and all data is backed-up daily to S3.

FSx for Windows File Server **can be joined to an existing on-premises Active Directory**. This allows the file system to continue using the same AD groups and policies to restrict access to the shares, folders, and files, just as it did on-premises.

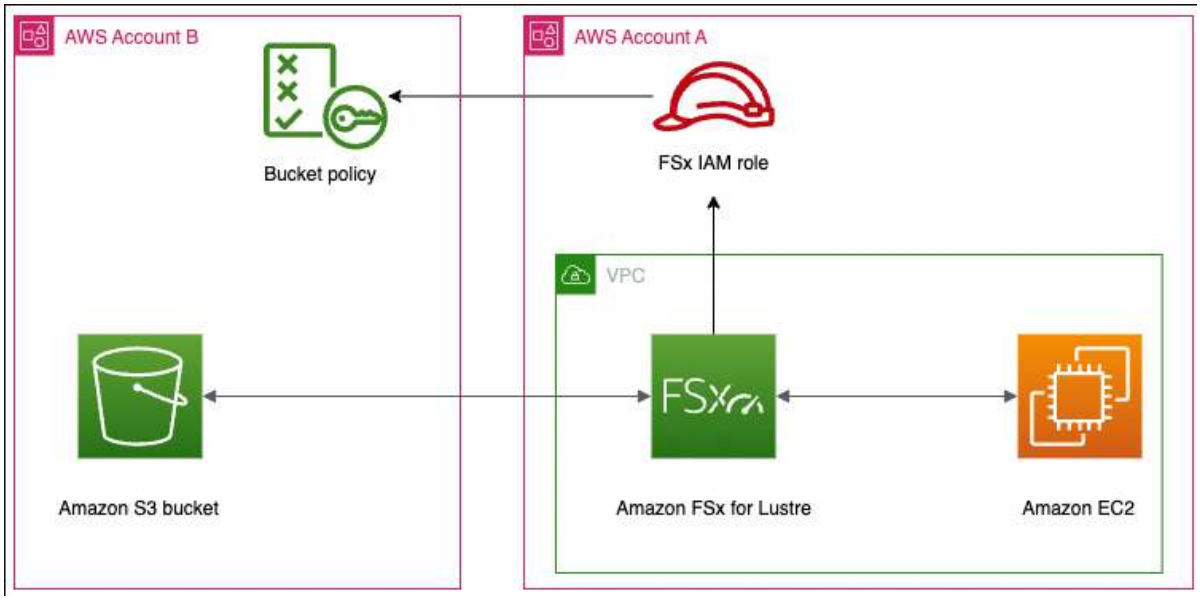


▼ ☑ **FSx - Lustre.**

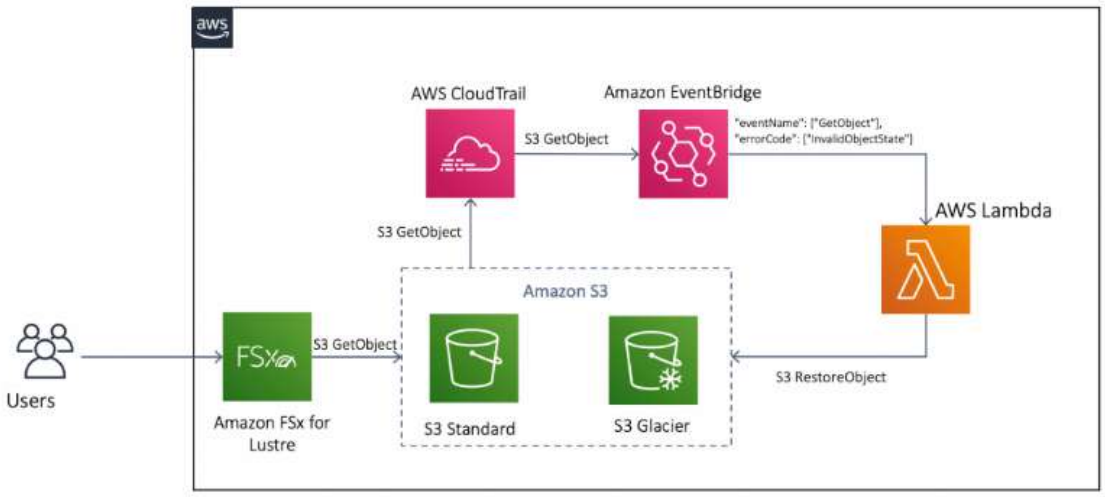
FSx for Lustre - type of **parallel** distributed file system, for large-scale computing. It is commonly used for HPC and uses ML. We can chose between SSD and HDD storage options. Can be used from on-premises servers (VPN or Direct Connect).

FSx for Lustre uses a striped architecture, where data is distributed across multiple disks to optimize performance.

We can read S3 as a file system through FSx and write the output of the computations back to S3 (through FSx).



▼ **Solution Architecture - Automate File Retrieval from S3 Glacier.**

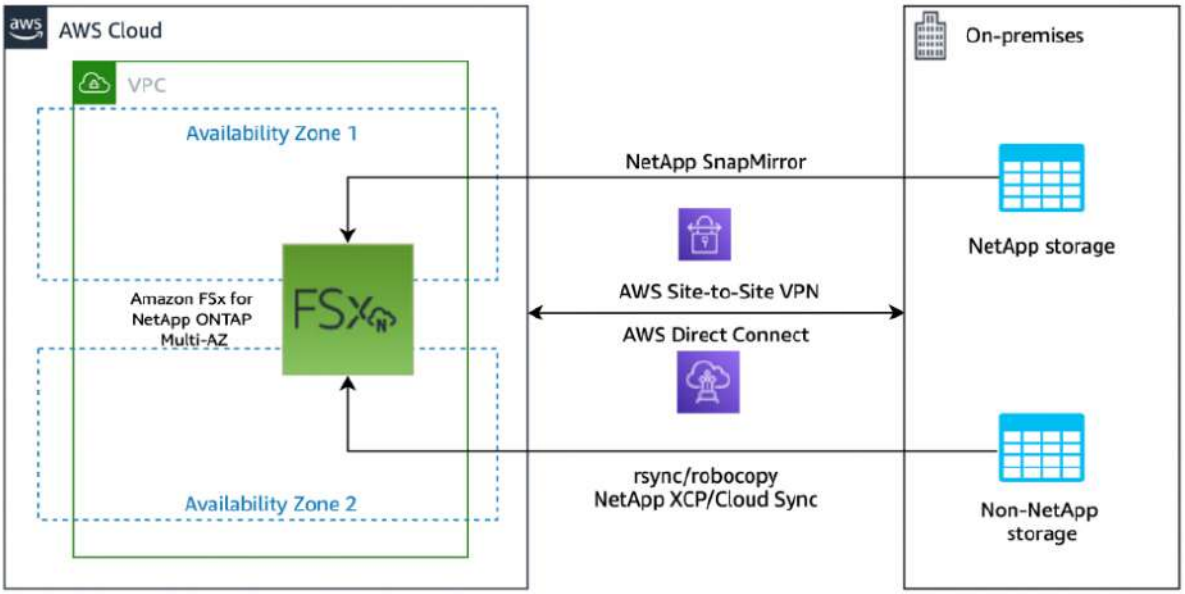


▼ ☑ **FSx - NetApp ONTAP & OpenZFS.**

FSx for NetApp ONTAP - managed NetApp ONTAP on AWS. It moves workloads running on ONTAP or NAS to AWS. It is compatible with NFS, SMB, iSCSI protocol.

Its storage shrinks or grows automatically and we can do snapshots, replication, data de-duplication and compression. We can do point-in-time instantaneous cloning (helpful for testing new workloads).

ONTAP SnapMirror - a replication technology in ONTAP that allows us to asynchronously replicate data between two storage systems.



FSx for OpenZFS - managed OpenZFS file system on AWS. It is compatible with NFS and moves workloads running on ZFS to AWS. We can do point-in-time instantaneous cloning. Also can do snapshots and compression.

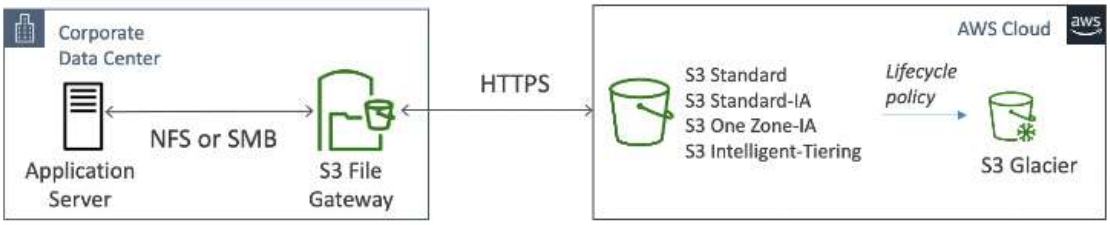
▼ ☑ **AWS Storage Gateway.**

S3 is a proprietary storage technology and only way to expose S3 data on-premise is using AWS Storage Gateway.

AWS Storage Gateway is a bridge between on-premise data and cloud data in S3. Our hybrid storage service will allow on-premises to seamlessly use the AWS Cloud (disaster recovery, backup & restore, tiered storage).

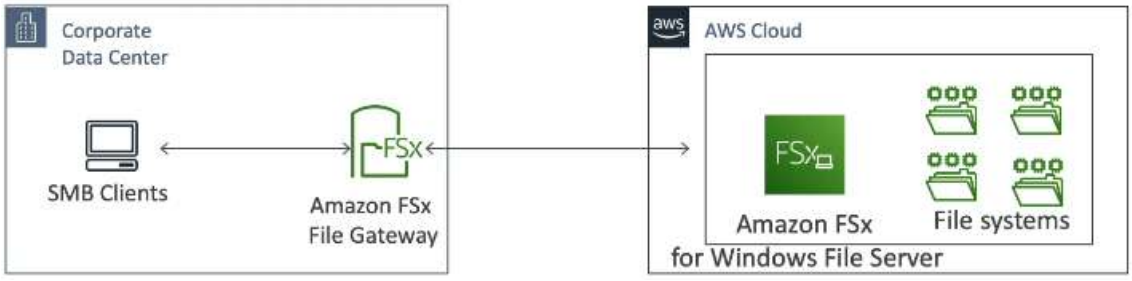
Types of Storage Gateway:

- **S3 File Gateway** - offers file-based access to objects in Amazon S3. It allows us to store and retrieve objects in Amazon S3 using file protocols, such as **NFS and SMB**. Most recently used data is cached in the file gateway. It supports all S3 classes except S3 Glacier. Transition to S3 Glacier must be done using a Lifecycle Policy.



To access our buckets we must create IAM role for each File Gateway. Also if we use SMB we can use its integration with Active Directory for user authentication.

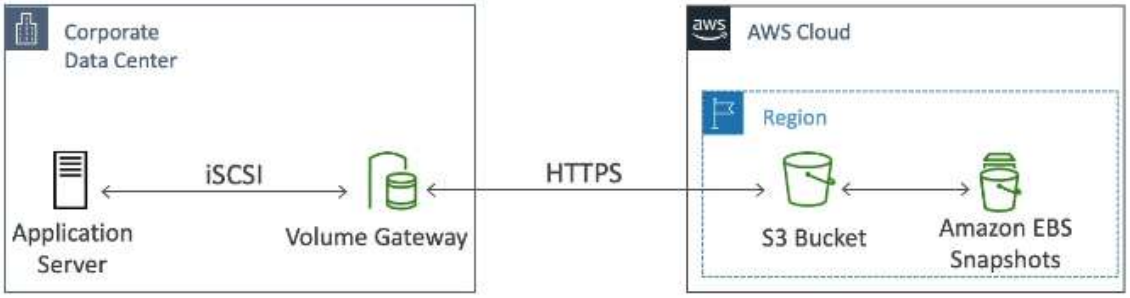
- **FSx File Gateway** - offers native access to Amazon FSx for Windows File Server. It has local cache for frequently accessed data.



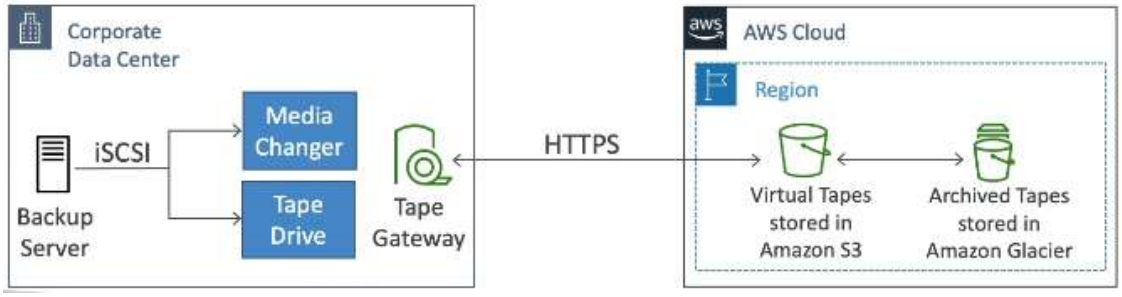
- **Volume Gateway** - provides block storage to on-premises applications using iSCSI protocol, with data asynchronously backed up to AWS.

  There are two modes for Volume Gateway:

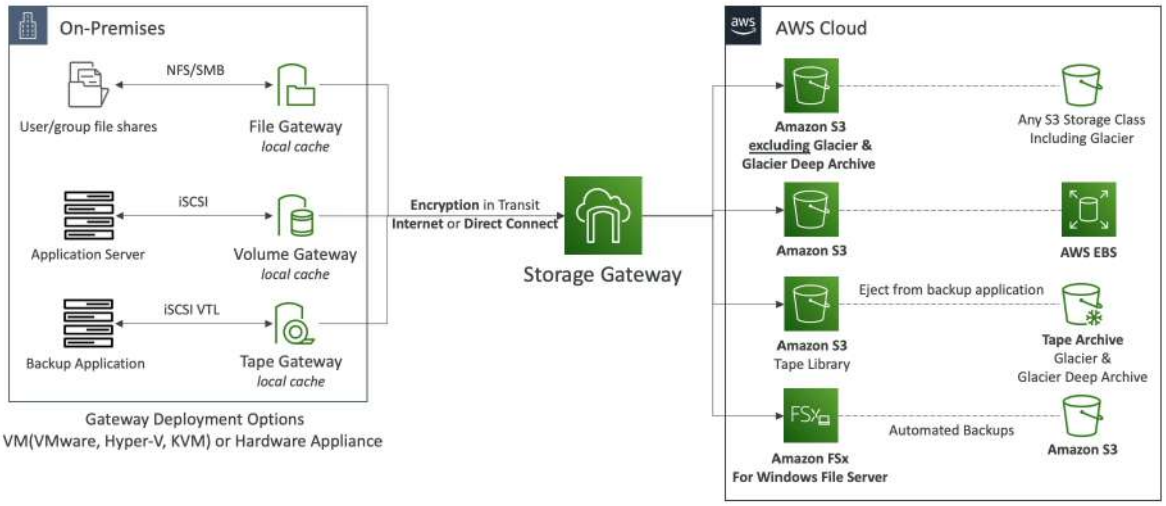
  - **Cached Volumes** - frequently accessed data is cached locally, while the primary data is stored in S3.
  - **Stored Volumes** - entire datasets are stored locally, with scheduled backups to S3.



- **Tape Gateway** - enables us to use Amazon S3 and Glacier as a scalable, cost-effective solution for archival and backup data stored on physical tapes. It presents a virtual tape library (VTL) interface, compatible with leading backup software.



Using Storage Gateway means we need on-premises virtualization. Otherwise, we can use a Storage Gateway Hardware Appliance. It has the required CPU, memory, network and SSD cache resources.



▼ ☑ **AWS Transfer Family.**

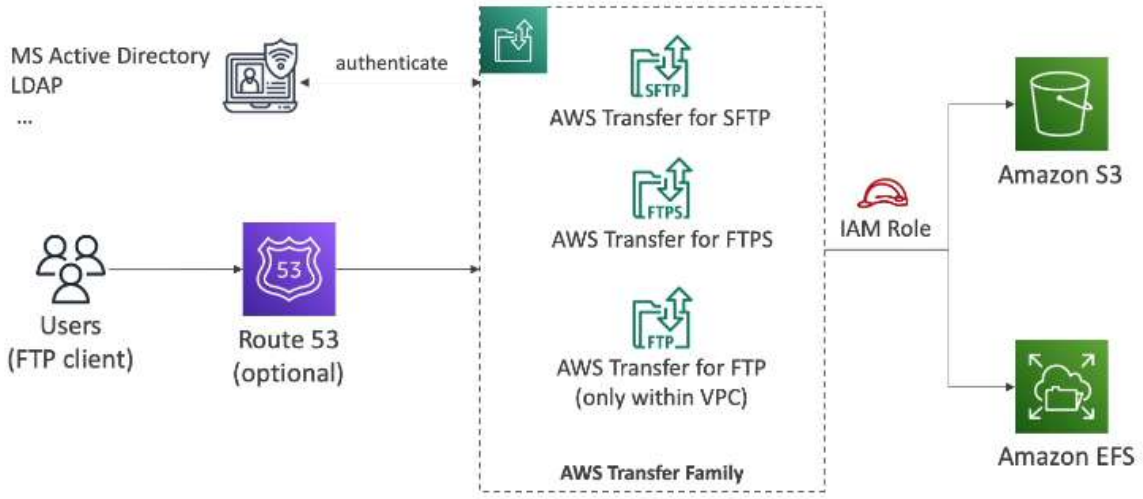
Transfer Family is a set of fully managed services for file transfers into and out of S3 or EFS using the FTP protocol. We pay per provisioned endpoint per hour and data transfers in GB. It is possible to store and manage users' credentials within the service.

- **AWS Transfer for FTP**
- **AWS Transfer for FTPS**
- **AWS Transfer for SFTP**

Can be integrated with existing authentication systems.

With Transfer Family's AS2 capabilities, we can securely exchange AS2 messages at scale while maintaining compliance and interoperability with trading partners. **Applicability Statement 2 (AS2)** is a business-to-business (B2B) messaging protocol used to exchange Electronic Data Interchange (EDI) documents.

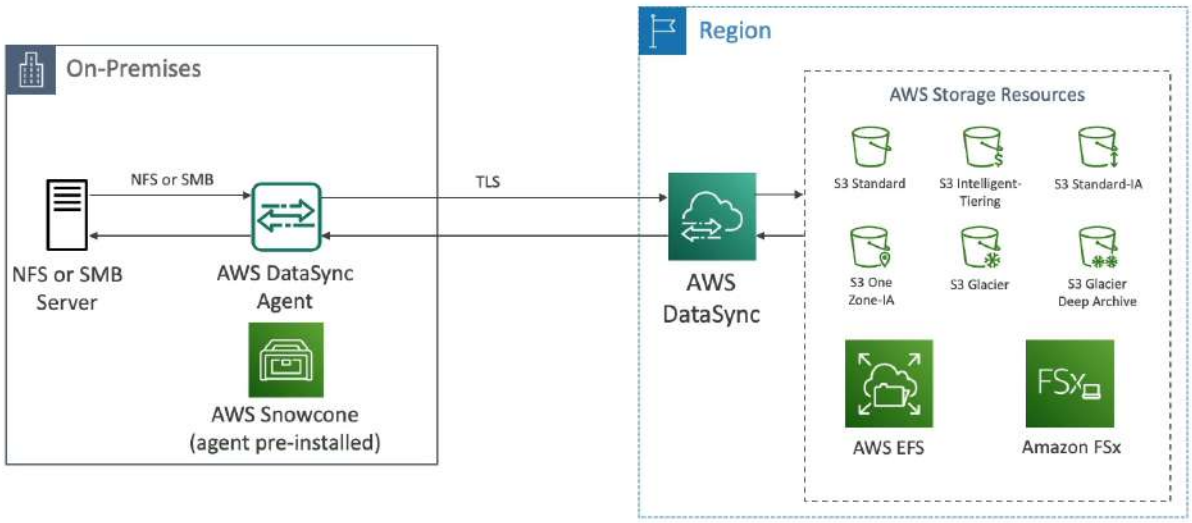
Use cases: sharing files, public datasets, CRM, ERP (Enterprise Resource Planing).



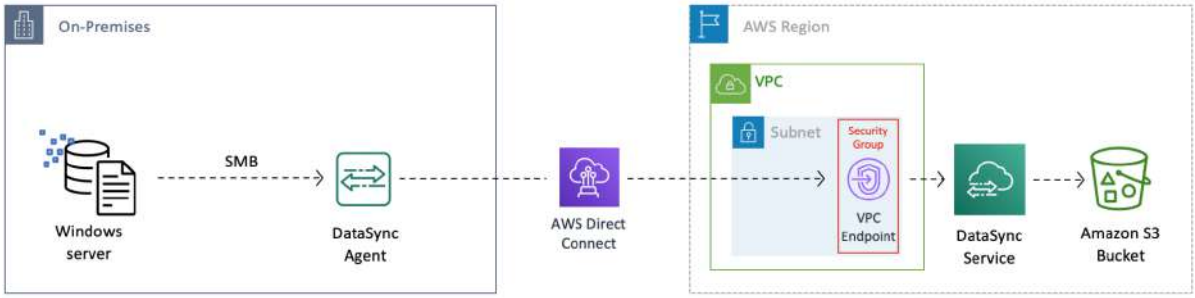
▼ ☑ **AWS DataSync.**

DataSync is used to automate and accelerate the replication of data and move large amount of data to and from on-premises to AWS (need agent) or AWS to AWS. It can synchronize to **S3, EFS** and **FSx**. Replication tasks can be scheduled hourly, daily or weekly.

File permissions and metadata are preserved (NFS POSIX, SMB...). One agent task can use 10 Gbps and we can setup a bandwidth limit.



The DataSync agent can route DataSync traffic from the on-premises storage system (source location) to the DX connection.



Decoupling Applications

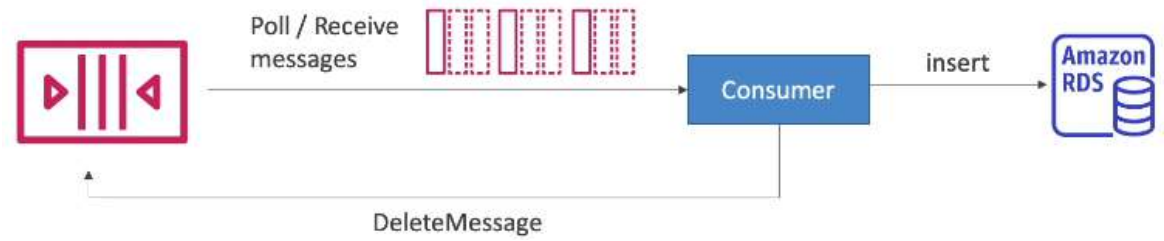
▼ ☑ **AWS SQS - Standard Queues.**

SQS is a fully managed message queuing service that enables us to decouple and scale microservices, distributed systems, and serverless applications.

There is no limit how many messages can be in the queue. Default retention of messages is 4 days, maximum of 14 days. Messages in the SQS queue will continue to exist even after the EC2 instance has processed it, until we delete that message. Consumers share the work to read messages and scale horizontally. It can have duplicate messages.

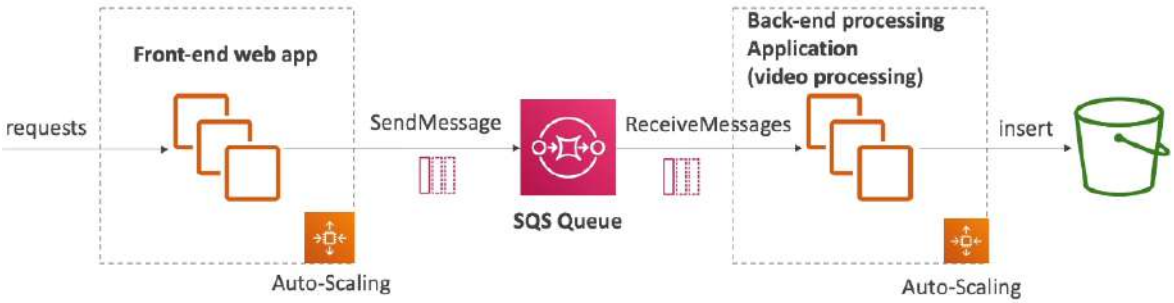
We can use SendMessage API by AWS SDK to produce messages. The message is **persisted** in SQS until a consumer deletes it.

Message consumers are running on EC2 instances, servers or AWS Lambda. They poll SQS for messages, process the messages and then delete them using DeleteMessage API.



Consumers receive and process messages in parallel and we can scale consumers horizontally to improve throughput of processing.

Decouple between Application Tiers



SQS Security

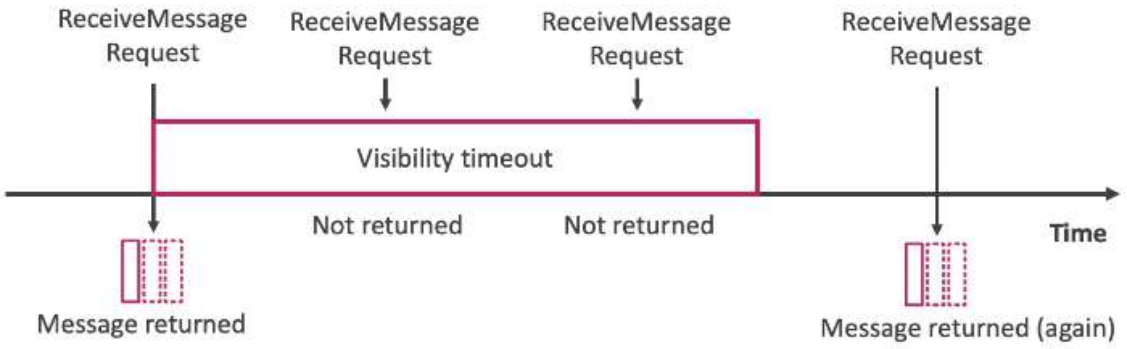
Encryption - has in-flight encryption using HTTPS, at-rest encryption using KMS and we can do client-side encryption.

Access Controls - we can configure IAM policies to regulate access to the SQS API.

SQS Access Policies - similar to S3 bucket policies. Useful for cross-account access to SQS queues. They allow other services to write to an SQS queue.

▼ ☑ **SQS Message Visibility Timeout.**

After a message is polled by a consumer, it becomes invisible to other consumers. By default, the message visibility timeout is 30s (max 12h). After the message visibility timeout is over, the message is visible in SQS.

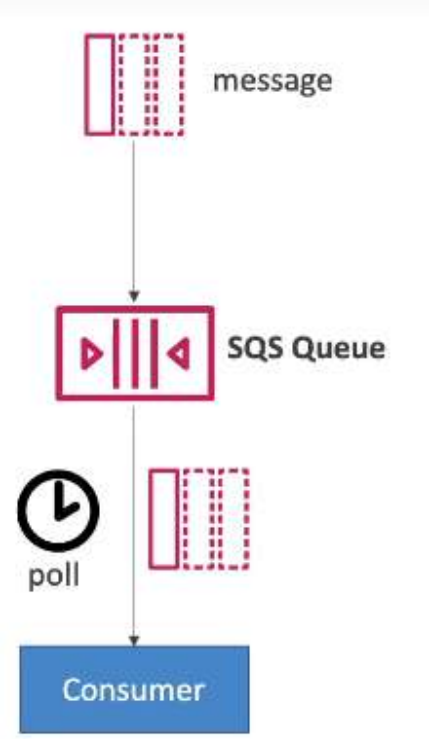


If a message is not processed within the visibility timeout, it will be processed twice. A consumer could call the ChangeMessageVisibility API to get more time.

If visibility timeout is high (hours), and consumer crashes, re-processing will take time and if visibility timeout is too low (seconds), we may get duplicates.

▼ ☑ **SQS Long Polling.**

Long Polling - when a consumer requests messages from the queue, it can optionally wait for messages to arrive if there are none in the queue.



Long Polling decreases the number of API calls made to SQS while increasing the efficiency and latency of our application. The **wait time can be between 1-20 seconds**.

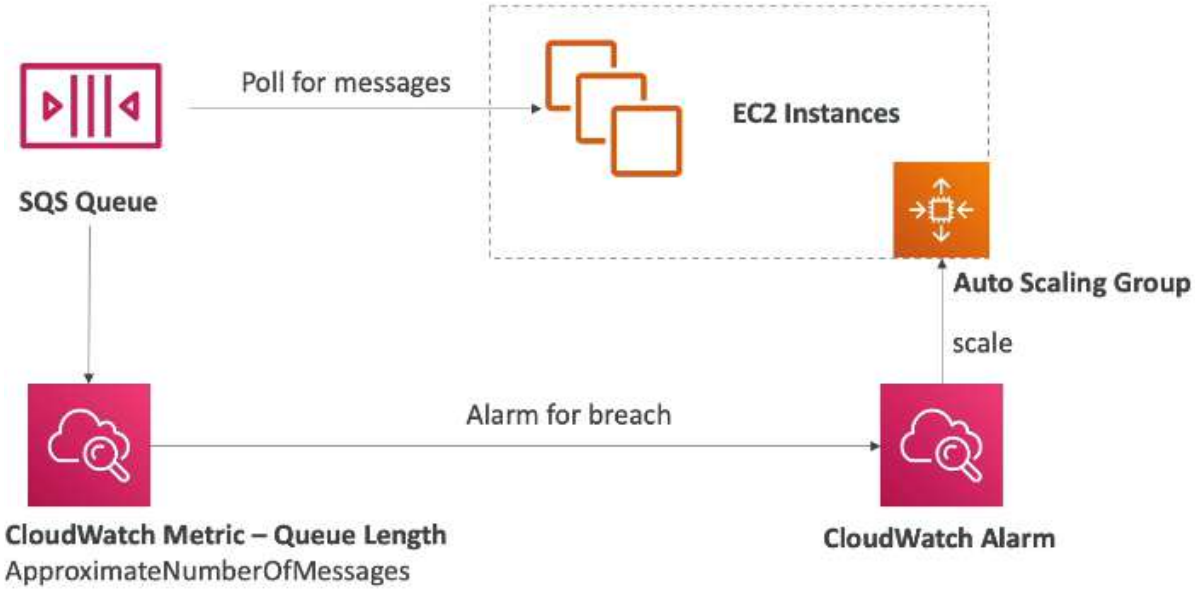
It can be enabled at the queue level or at the API level using **WaitTimeSeconds**.

▼ ☑ **AWS SQS - FIFO Queues.**

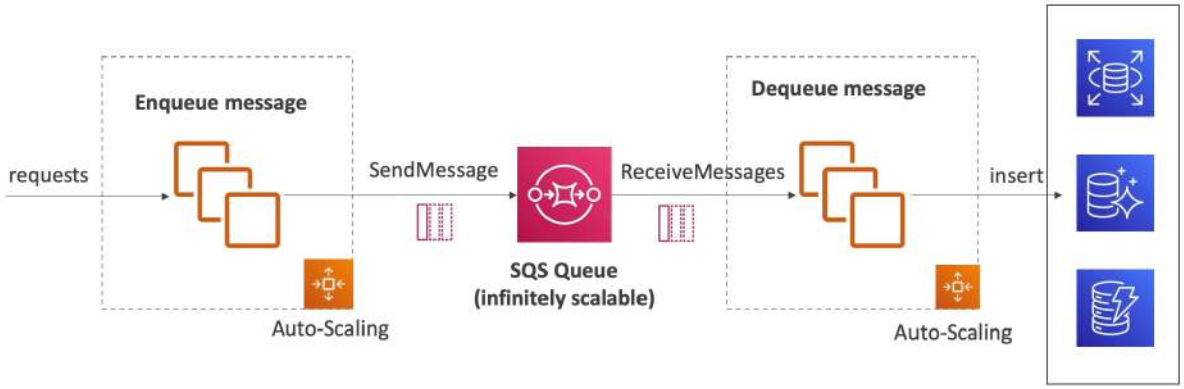
In Standard SQS messages are processed in random order by the consumer. If we want ordering of messages in the queue there is **SQS FIFO Queue** variant. There messages are processed in order by the consumer.

▼ ☑ **SQS + ASG Design Patterns.**

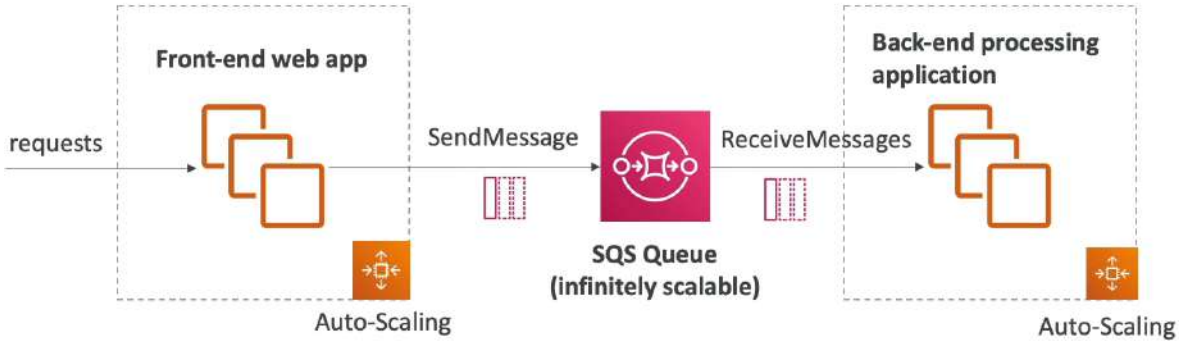
Scale Based on CloudWatch Alarm



Buffer to Database Writes

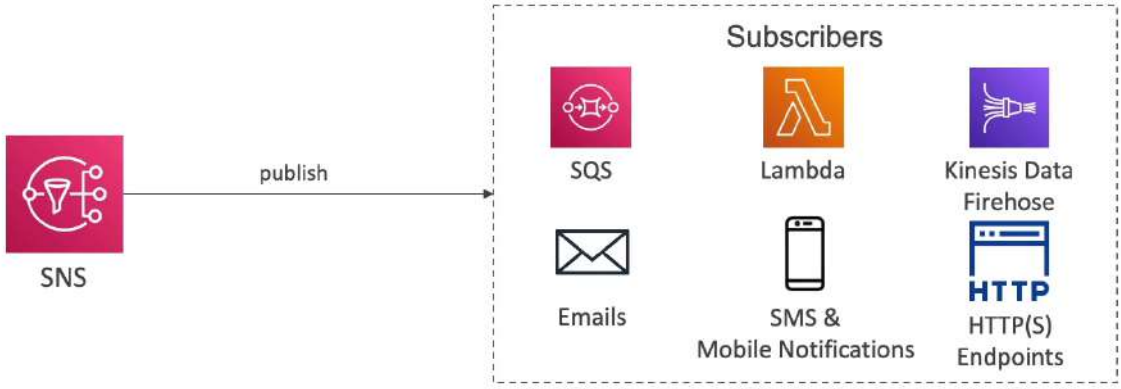


Decouple between Application Tiers



▼ ☑ **AWS SNS.**

SNS is a topic-based publisher/subscriber model. The event publishers only sends message to one SNS topic. Subscribers receive messages by subscribing to a topic. Subscribers can be various endpoints such as email, SMS, AWS Lambda functions. Each subscriber to the topic will get all the messages.



- **Topic Publish** - using SDK. First create topic, then create a subscription and publish to the topic.
- **Direct Publish** - for mobile apps SDK. We need to create a platform application and platform endpoint. Then publish to the platform endpoint.

SNS Message Filtering - assign a filter policy to the topic subscription, and the subscriber will only receive a message that they are interested in.

SNS has **FIFO Topic** (ordering of messages in the topic). It has similar features as SQS FIFO:

- **Ordering** by Message Gorup ID.

- **Deduplication** using a Deduplication ID or Content Based Deduplicaton.

Both SQS Standard and FIFO queues can be subscribers.

SNS Security

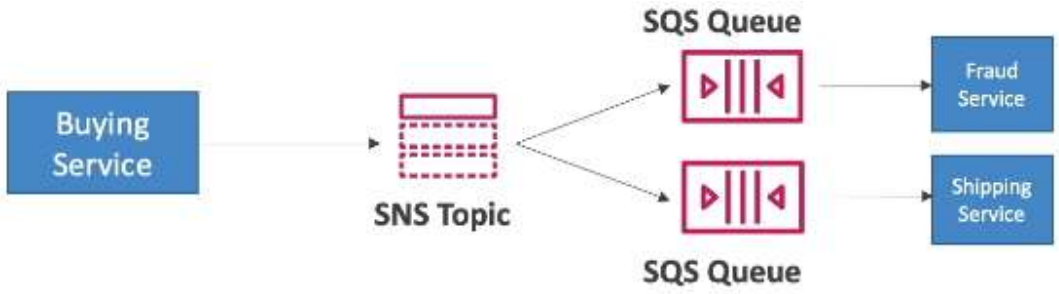
Encryption - has in-flight encryption using HTTPS, at-rest encryption using KMS and we can do client-side encryption.

Access Controls - we can configure IAM policies to regulate access to the SNS API.

SNS Access Policies - similar to S3 bucket policies. Useful for cross-account access to SNS topics. They allow other services to write to an SNS topic.

▼ ☑ SNS & SQS - Fan Out Pattern.

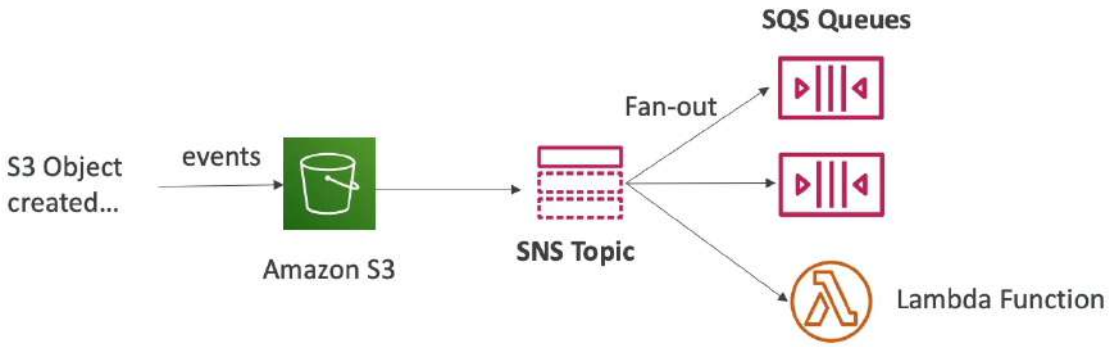
In the **Fan Out** pattern, we use SNS to broadcast messages to multiple SQS queues. Each queue can then be processed independently by separate consumers or applications. Need to configure SQS queue access policy to allow SNS to write.



It's fully decoupled and we dont have data loss. We have ability to add more SQS subscribers over time. It works with SQS queues in other regions.

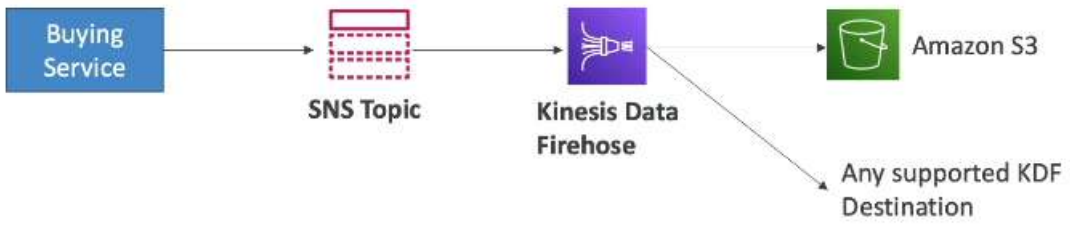
▼ Solution Architecture - S3 Events to Multiple Queues.

For the same combination of event type and prefix we can only have one S3 Event rule. If we want to send the same S3 event to many SQS queues we should use fan out.



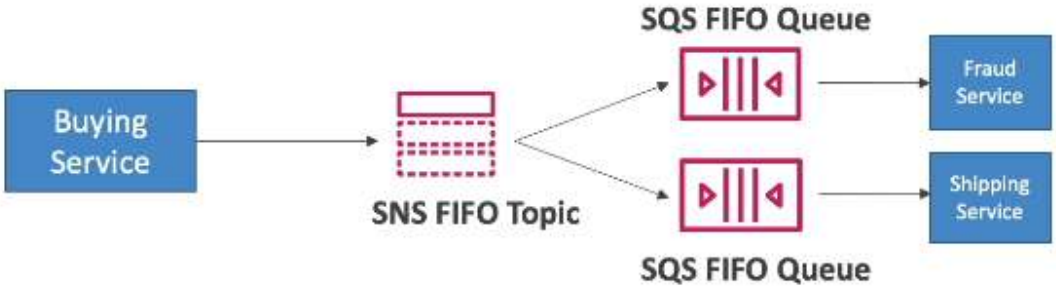
▼ **Solution Architecture - SNS to S3 through KDF.**

SNS can send to Kinesis and we can have the following solutions architecture:



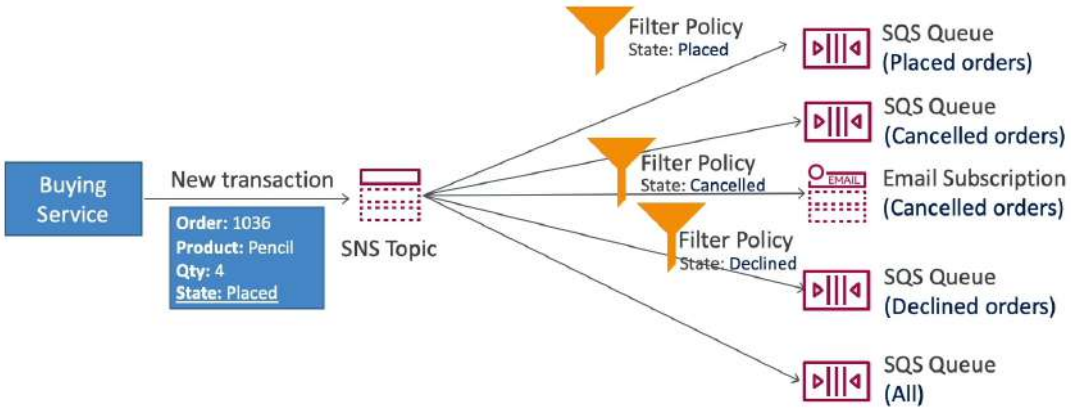
▼ **Solution Architecture - SNS FIFO + SQS FIFO Fan Out.**

In case we need fan out + ordering + deduplication.



▼ **Solution Architecture - Message Filtering.**

JSON policy is used to filter messages sent to SNS topic's subscriptions. If a subscription doesn't have a filter policy, it receives every message.



▼ ☑ **Amazon Kinesis.**

Kinesis is a managed service which collect, process and analyze real-time streaming data at any scale.

Kinesis components:

- **Kinesis Data Streams (KDS)** - collect and process large streams of data records in real time.
- **Kinesis Data Firehose** - loads streaming data into data lakes, data stores, and analytics services.
- **Kinesis Data Analytics** - analyzes streaming data using SQL or Apache Flink.
- **Kinesis Video Streams** - streams and process video in real time.

▼ ☑ **Kinesis Data Streams (KDS).**

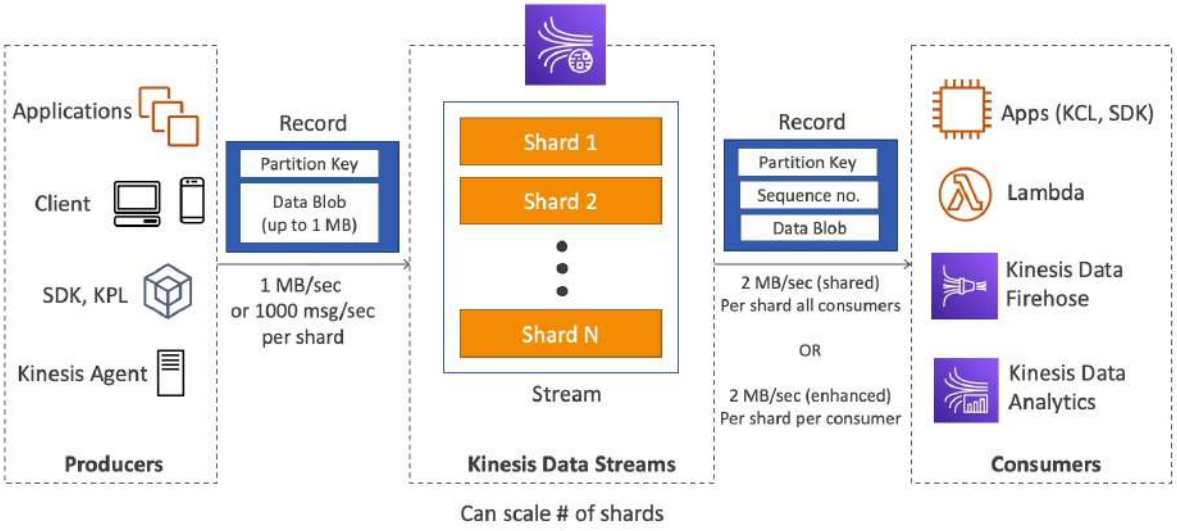
KDS is a real-time data streaming service designed to handle massive streams of data in real-time. We have ability to reprocess data. Once data is inserted in Kinesis, it cant be deleted.

Key Concepts

- **Stream** - a collection of records that are distributed across multiple shards. Each stream has a sequence of data records.
- **Shard** - the fundamental unit of capacity within a stream. A stream can have multiple shards, allowing for scaling up.
- **Record** - a unit of data in a stream. It consists of:
  - Data Blob - data we want to store.
  - Partition Key - string used to group records by shard within a stream.

    - Sequence Number - an identifier assigned to each record, unique per shard, that is used to maintain order.

- **Producers** - clients or services that write data to the KDS (AWS SDK, KPL, Kinesis Agent).
- **Consumers** - applications or services that process or consume the data (AWS SDK, KCL, AWS Lambda, KDF, KDA).



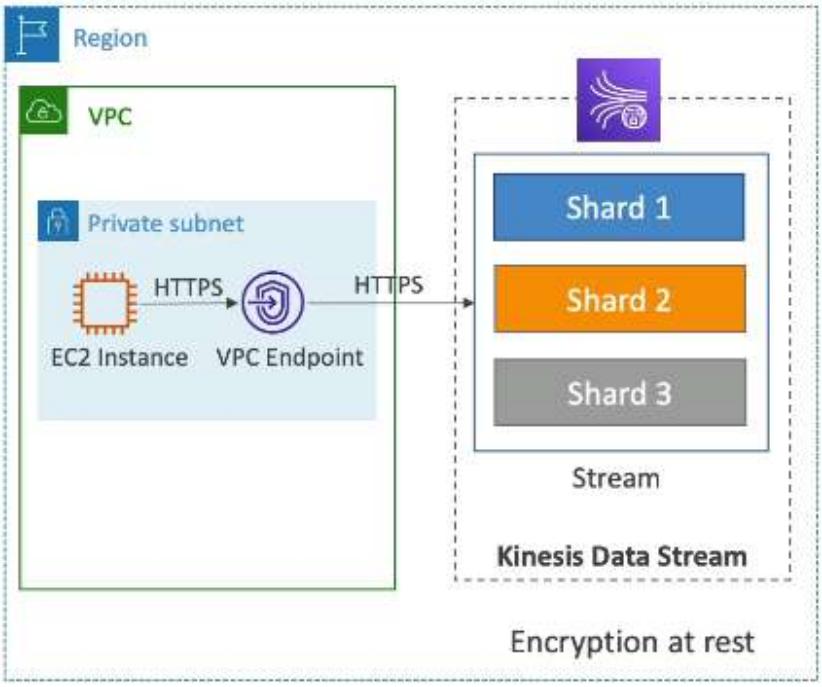
KDS Capacity Modes

- **Provisioned mode** - we choose the number of shards provisioned, scale manually or using API. We pay per shard provisioned per hour.
- **On-demand mode** - no need to provision or manage the capacity. Scales automatically based on observed throughput peak during the last 30 days. We pay per stream per hour & data in/out per GB.

KDS Security

We can control access/authorization using IAM policies. Encryption in flight is done using HTTPS endpoints and at rest using KMS. Also we can implement encryption/decryption of data on client side.

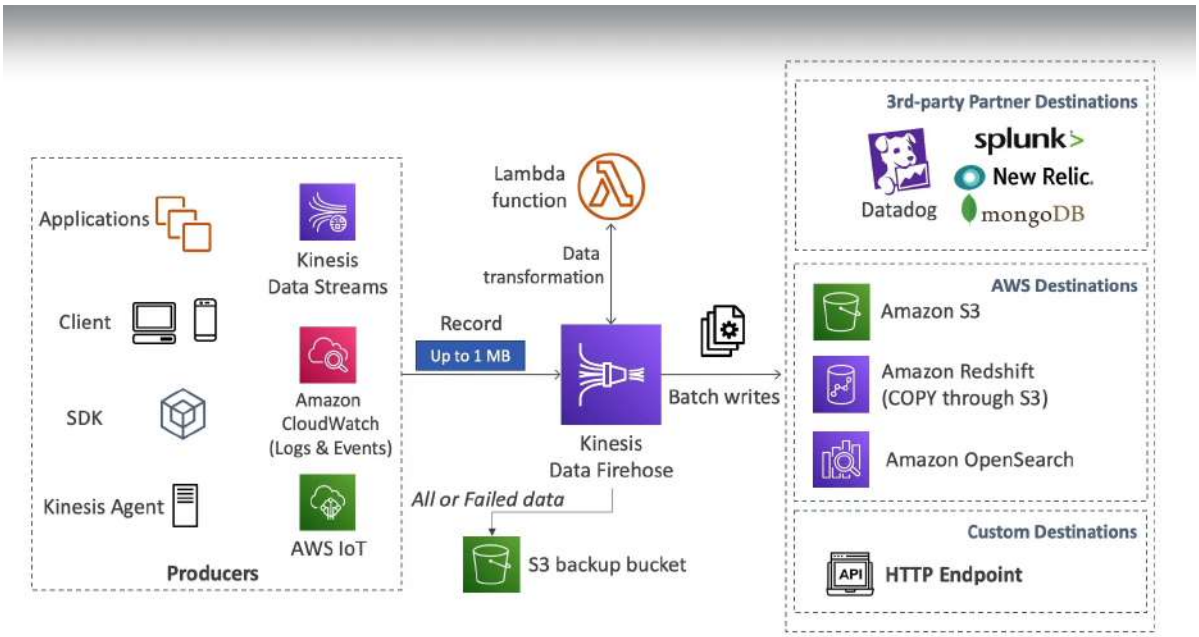
We can use VPC Endpoints for Kinesis to access within VPC. To monitor API calls we use CloudTrail.



▼ ☑ **Kinesis Data Firehose (KDF).**

KDF is a fully managed service that makes it easy to reliably load streaming data into data lakes, data stores, and analytics services. We pay for data going through Firehose.

It supports many data formats, conversions, transformations and compresion. It supports custom data transformations using AWS Lambda. Firehose can send failed or all data to a backup S3 bucket.



Because we write data in batches, KDF is near real time. If we set buffer interval to zero seconds (no buffering) it will be still considered near real time.

Kinesis Data Streams	Kinesis Data Firehose
• Streaming service for ingest at scale • Write custom code (producer / consumer) • Real-time (~200 ms) • Manage scaling (shard splitting / merging) • Data storage for 1 to 365 days • Supports replay capability	• Load streaming data into S3 / Redshift / OpenSearch / 3rd party / custom HTTP • Fully managed • Near real-time • Automatic scaling • No data storage • Doesn't support replay capability

▼ ☑ **Kinesis Data Analytics (KDA).**

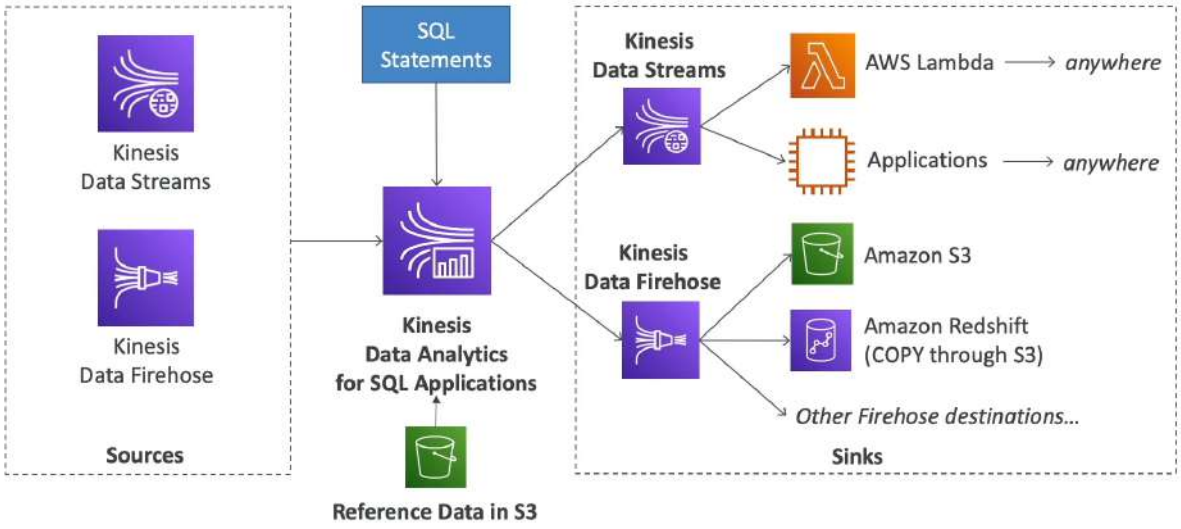
KDA for SQL Applications

It provides real-time analytics on KDS & KDF using SQL. We can add reference data from S3 to enrich streaming data. It is fully managed, no servers to provision and has automatic scaling.

Output:

- KDS - create streams out of real-time analytics query.
- KDF - send analytics query results to destinations.

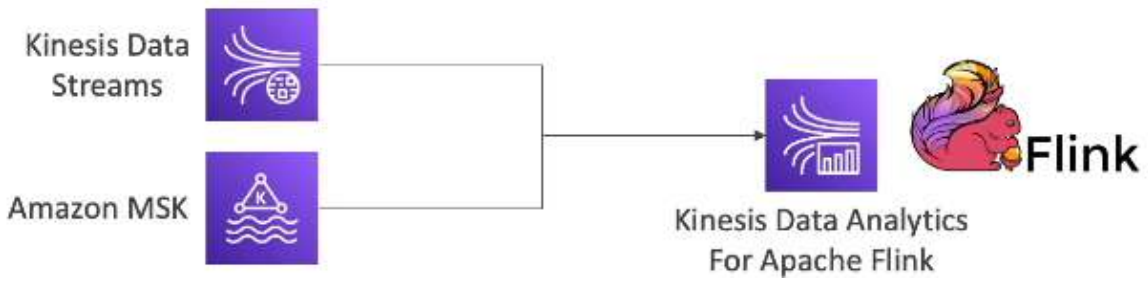
Use cases: time-series analytics, real-time dashboards, real-time metrics.



KDA for Apache Flink

Supports building and running Apache Flink applications. We get automatic provisioning compute resources, parallel computation and automatic scaling and automatic backups. Flink does not read from KDF.

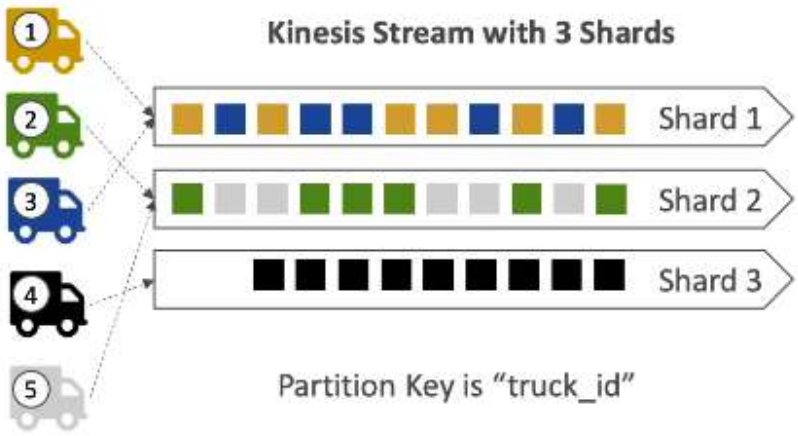
> Flink is a powerful open-source stream processing framework that provides advanced capabilities for managing state, fault tolerance, and complex event processing.



▼ ☑ **Data Ordering for KDS & SQS FIFO.**

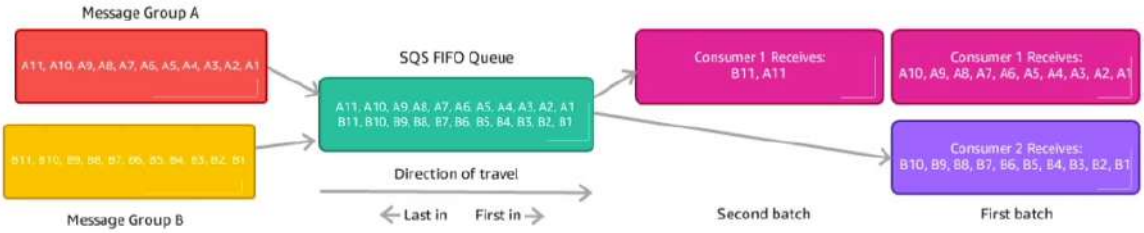
To ensure ordering for specific types of data in KDS, we can use partition keys. Events with the same partition key are routed to the same shard, and thus will be

read in order.



For SQS FIFO, if we dont use a Group ID, messages are consumed in the order they are sent, with only one consumer.

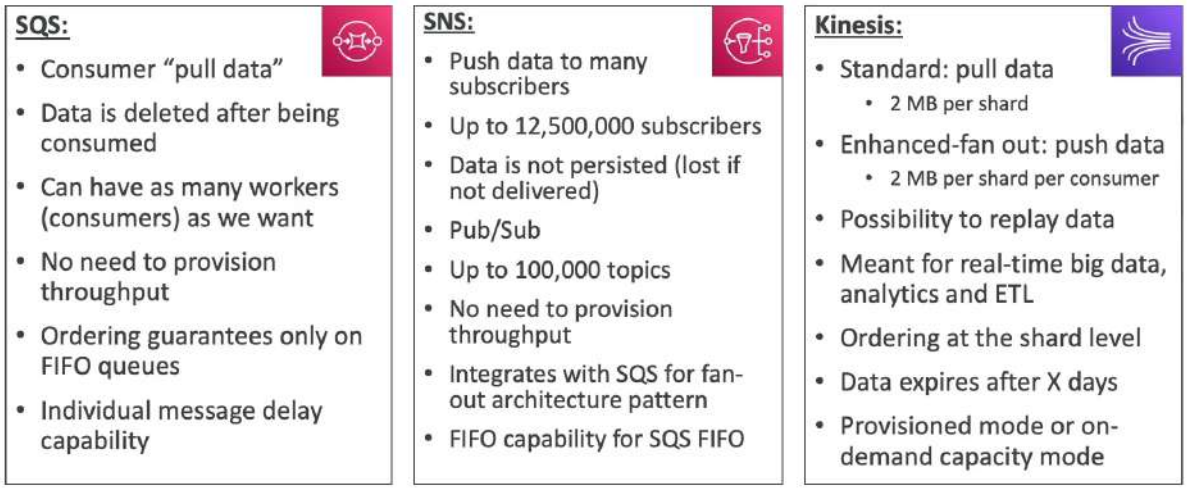
If we want to scale the number of consumers, but want messages to be "grouped" when they are related to each other, then we should use a Group ID (similar to Partition Key in Kinesis).



Lets assume 100 trucks, 5 kinesis shards, 1 SQS FIFO:

- Kinesis Data Streams:
  - On average you'll have 20 trucks per shard
  - Trucks will have their data ordered within each shard
  - The maximum amount of consumers in parallel we can have is 5
  - Can receive up to 5 MB/s of data
- SQS FIFO
  - You only have one SQS FIFO queue
  - You will have 100 Group ID
  - You can have up to 100 Consumers (due to the 100 Group ID)
  - You have up to 300 messages per second (or 3000 if using batching)

☑ SQS vs SNS vs Kinesis.



☑ AWS SWF.

SWF (Simple Workflow Service) is a fully managed service that helps us coordinate and manage complex workflows. It allows us to build and run applications that coordinate tasks across distributed components or microservices.

- **Workflow** - series of steps or tasks that need to be executed in a specific order or with specific dependencies. SWF helps us manage the execution of these workflows, ensuring that tasks are completed and handling retries and failures.
- **Activity** - task that is performed as part of the workflow. Activities are typically implemented as code that performs the work and communicates

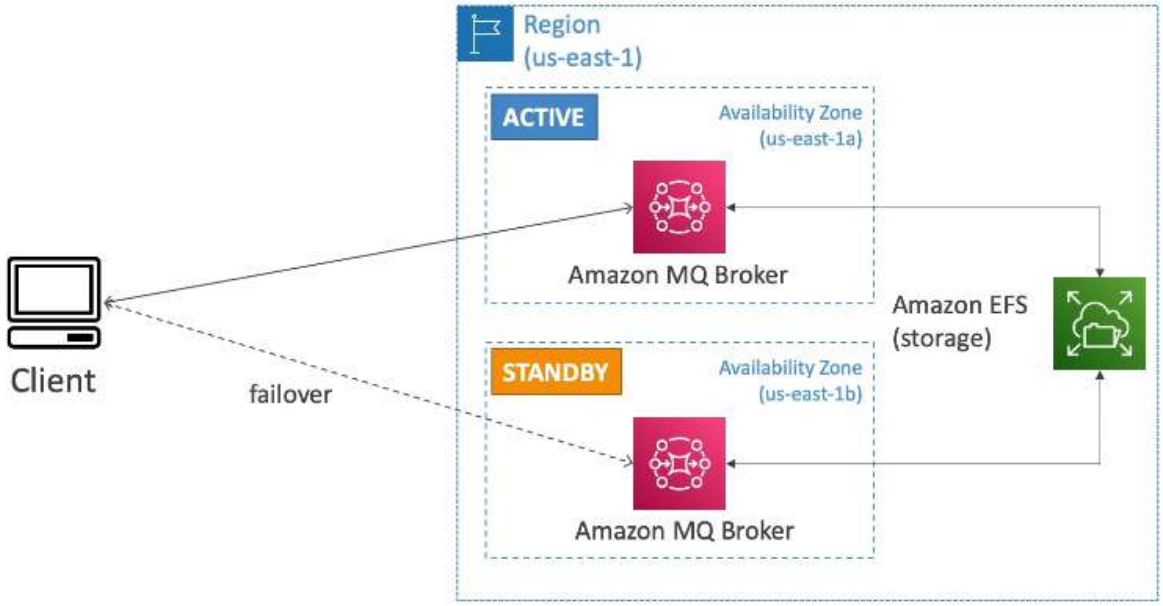
with external systems or services.

- **Task** - unit of work within an activity or a workflow. Tasks are sent to worker processes that perform the actual work.

▼ ☑ **Amazon MQ.**

When migrating to the cloud, instead of re-engineering the application to use SQS and SNS, we can use Amazon MQ. **Amazon MQ** is a managed message broker service for RabbitMQ and ActiveMQ.

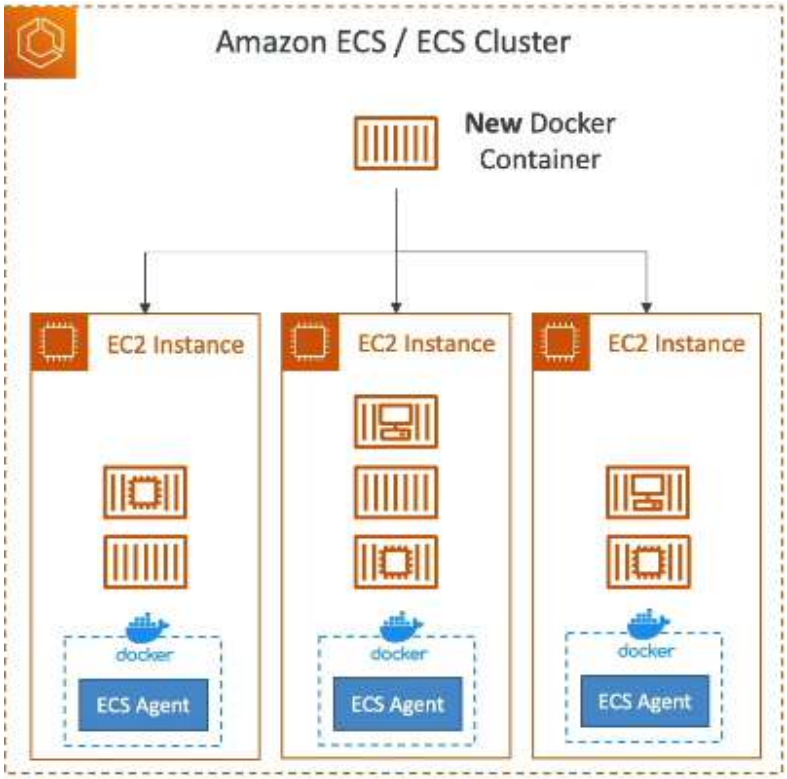
Amazon MQ doesn't scale as much as SQS and SNS. Runs on servers but, can run in Multi-AZ with failover and has both queue feature (like SQS) and topic features (like SNS).



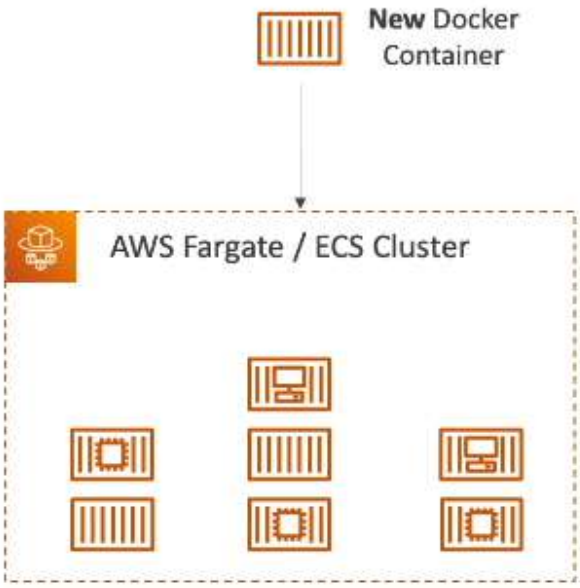
Containers on AWS

▼ ☑ **ECS & Fargate.**

ECS [EC2 Launch Type] - we must provision & maintain the infrasturcture (EC2 instances). Each EC2 Instance must run the ECS Agent to register in the ECS Cluster. AWS takes care of starting/stopping containers.

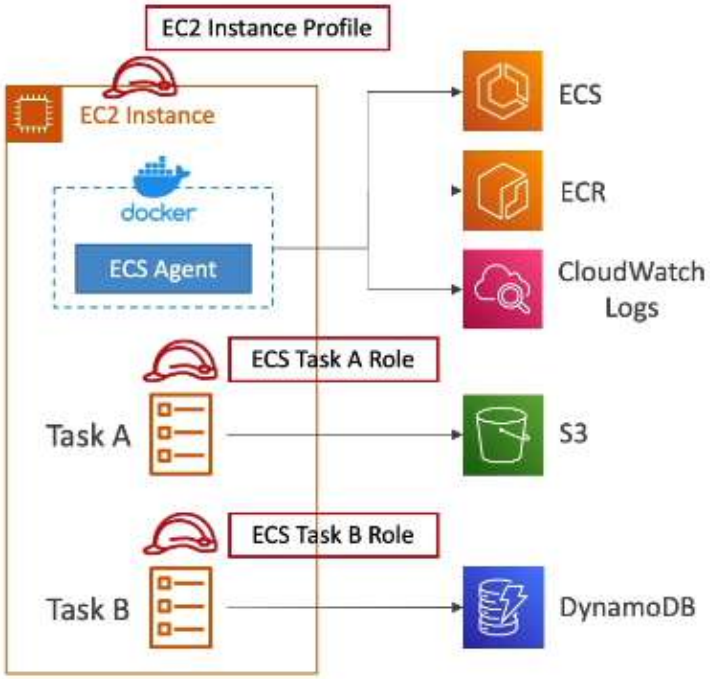


ECS [Fargate Launch Type] - we do not provision the infrasturcture, it's all serverless. We just create task definitions. AWS just runs ECS Tasks for us based on the CPU/RAM we need. To scale, we just need to increase the number of tasks (no more EC2 instances).



IAM Roles for ECS

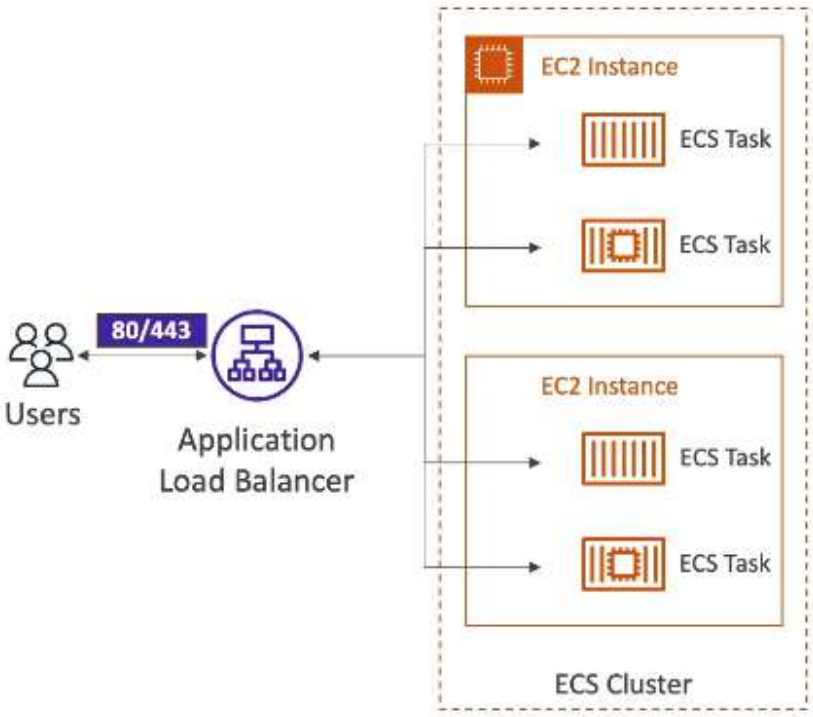
- **EC2 Instance Profile** (EC2 Launch Type only) - used by ECS agent to make API calls to ECS service. We can send container logs to CloudWatch Logs, pull Docker image from ECR, reference sensitive data in Secrets Manager or SSM Parameter Store.
- **ECS Task Role** - allows each task to have a specific role. Each role can use different ECS services.



ECS - ELB Integrations

ALB - supported and works for most use cases.

NLB - recommended only for high throughput/high performance use cases or to pair it with AWS Private Link.



ECS Data Volumes (EFS)

We can mount EFS file systems onto ECS tasks. It works for both EC2 and Fargate launch types.

Tasks running in any AZ will share the same data in the EFS file system. If we use Fargate + EFS it's totally serverless.

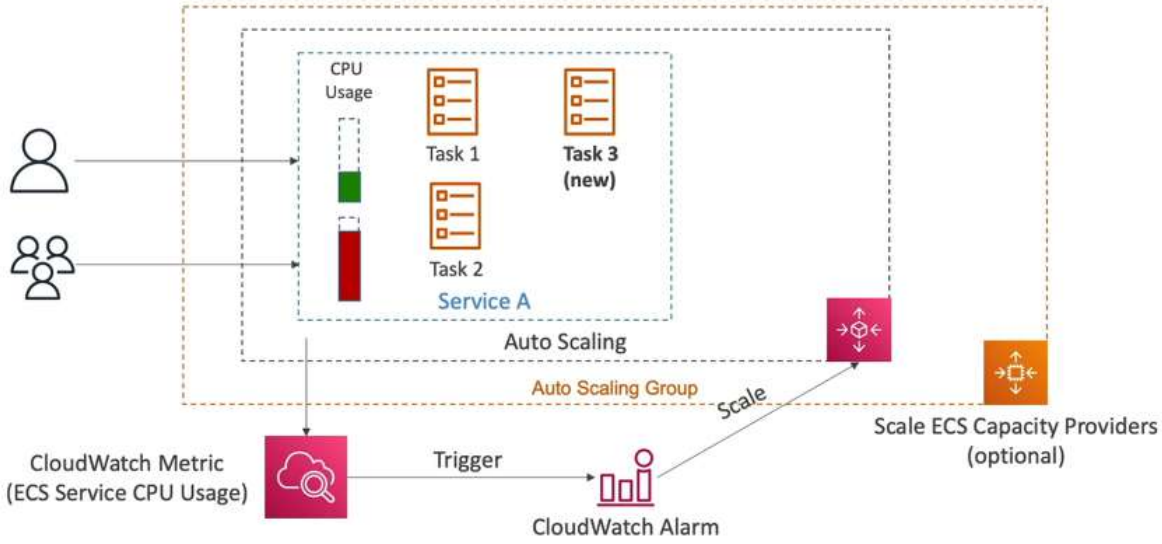
▼ ☑ **ECS Auto Scaling.**

With **ECS Auto Scaling** we can automatically increase/decrease the desired number of ECS tasks. Fargate Auto Scaling is much easier to setup because it is Serverless.

ECS Auto Scaling is similar but not equal to EC2 Auto Scaling (task level vs instance level).

- **Target Tracking Scaling** - scale based on target value for a specific CloudWatch metric.
- **Step Scaling** - scale based on a specified CloudWatch Alarm.
- **Scheduled Scaling** - scale based on a specified date/time (predictable changes).

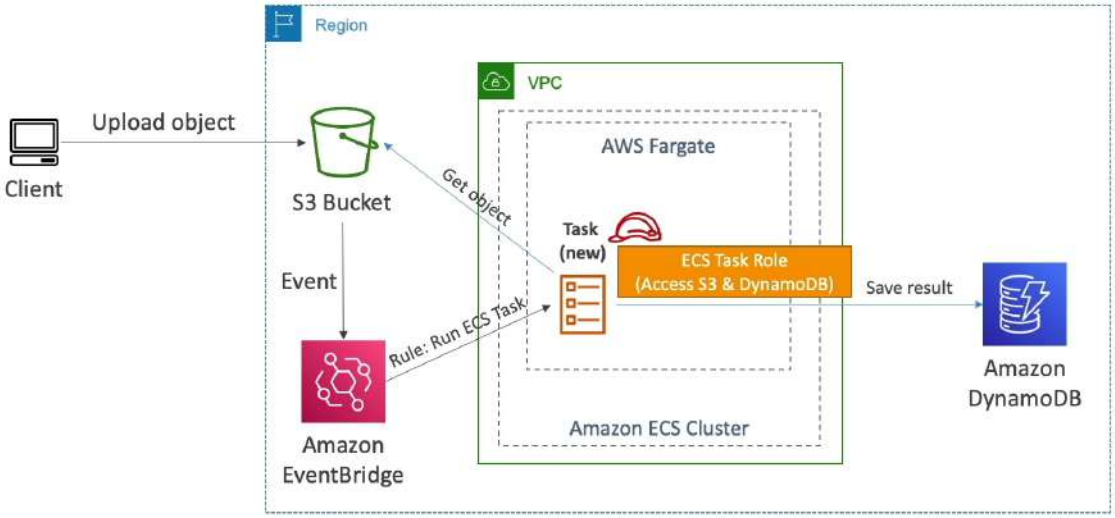
For EC2 launch type we can accommodate ECS scaling by adding underlying EC2 instances (using ASG or ECS Cluster Capacity Provider). ECS Cluster Capacity Provider is better.



▼ ☑ **ECS Solution Architectures.**

▼ **Tasks Invoked by Event Bridge.**

- Amazon EventBridge listens to events from AWS services, custom applications, or SaaS integrations.
- A rule is created in EventBridge that matches certain event patterns.
- When an event matching the rule occurs, EventBridge triggers ECS task or other service to process that event.
- For example, if an S3 object is created, an EventBridge rule can trigger a specific ECS Fargate task to process the file.



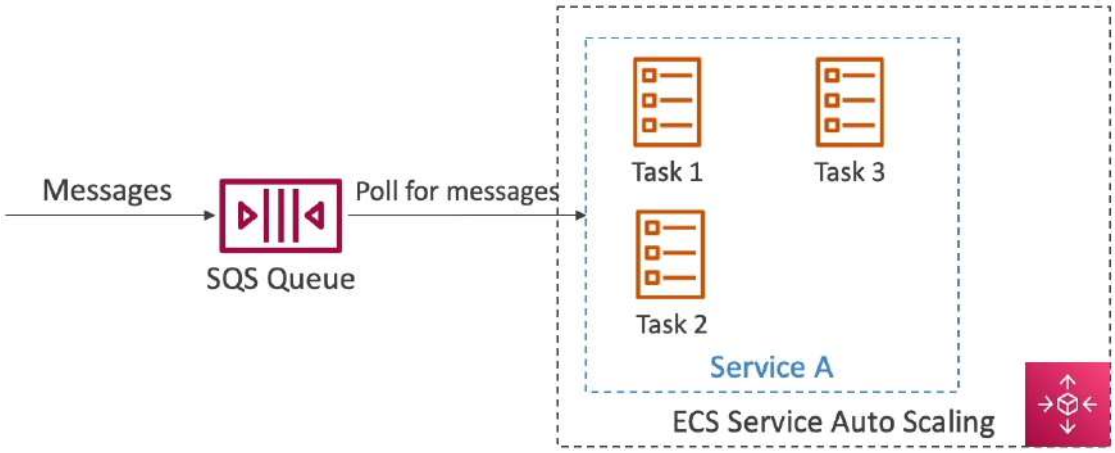
▼ Tasks Invoked by Event Bridge Schedule.

- EventBridge can be configured to run on a cron or rate expression (e.g. every hour).
- A scheduled rule in EventBridge invokes ECS tasks on the defined schedule.
- For example, we can run a scheduled ECS task every day at midnight to perform batch processing.

▼ ECS - SQS Queue.

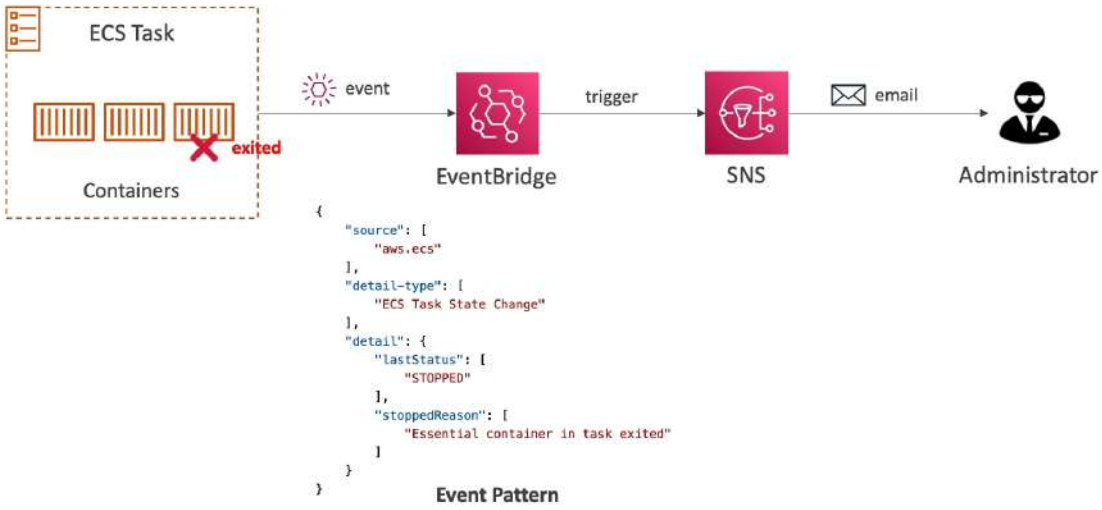
- Messages are sent to an SQS queue by producers, which can be other services, APIs, or applications.

- An ECS service with a task or set of tasks polls the SQS queue for messages.
- When a message is retrieved, the ECS task processes it. Once processed, the message is either deleted or moved to a dead-letter queue if it fails.



▼ **Intercept Stopped Tasks using Event Bridge.**

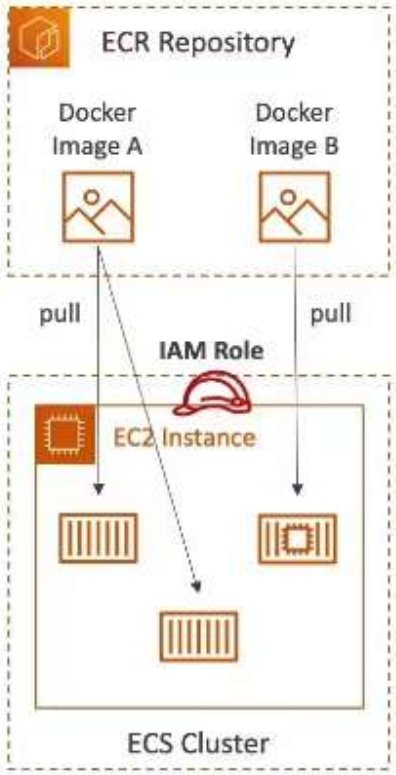
- ECS emits task state change events (e.g. task stopped, task started) which are sent to EventBridge.
- An EventBridge rule is set up to capture the “ECS Task State Change” events where the stopped status is detected.
- When a task stops, EventBridge triggers a target such as an SNS topic, Lambda function, or another ECS task for handling remediation, alerting, or cleanup actions.



▼ ☑ **ECR.**

ECR - private docker registry on AWS. It stores and manages Docker images on AWS. It is fully integrated with ECS and backed by S3.

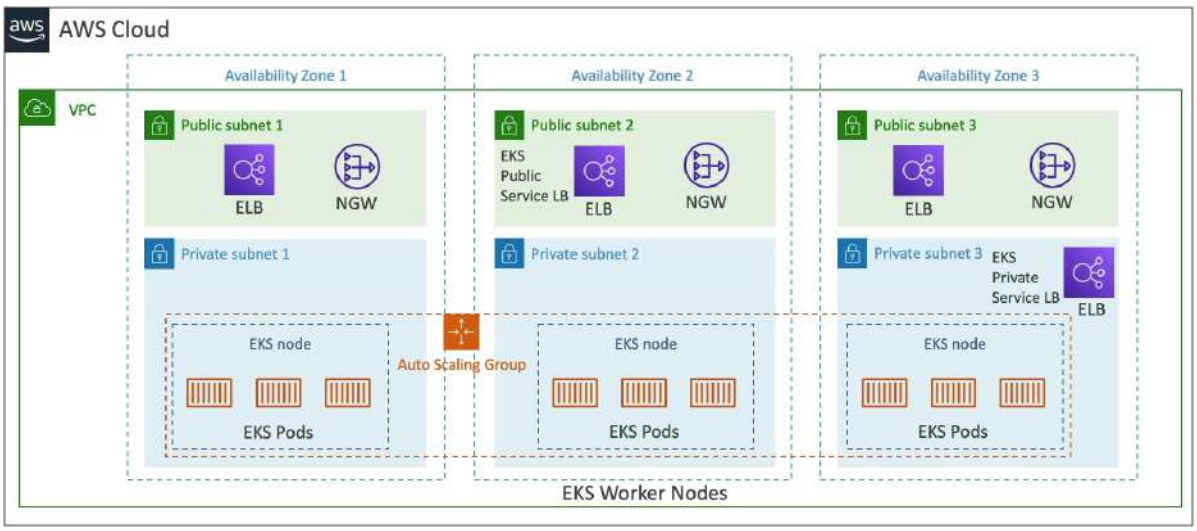
Access is controlled through IAM. It supports image vulnerability scanning, versioning, image tags, image lifecycle...



▼ ☑ EKS.

EKS is a managed service that simplifies the process of running Kubernetes on AWS. It is an alternative to ECS, similar goal but different API. Kubernetes is cloud-agnostic (can be used in any cloud).

EKS supports EC2 if we want to deploy worker nodes or Fargate to deploy serverless containers.



We can attach data volumes to our EKS cluster. We need to specify StorageClass manifest on our EKS cluster. It leverages a **Container Storage Interface (CSI)** compliant driver.

We have support for EBS, EFS (works with Fargate), FSx for Lustre, FSx for NetApp ONTAP.

Amazon EKS Connector is a feature that allows us to connect and manage Kubernetes clusters running outside of AWS (whether on-premises or in other cloud environments) within the Amazon EKS console.

Amazon EKS Anywhere is a deployment option for Amazon EKS that enables us to create and operate Kubernetes clusters on our own infrastructure, such as on-premises data centers or other cloud environments.

EKS Node Types

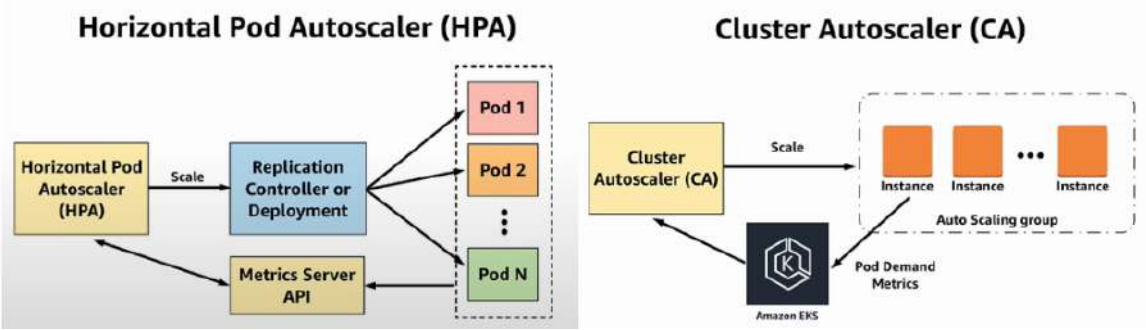
- **Managed Node Groups** - creates and manages Nodes (EC2 instances) for us. Nodes are part of an ASG managed by EKS. It supports on-demand or spot instances.

- **Self-Managed Nodes** - nodes created by us and registered to the EKS cluster and managed by an ASG. We can use prebuilt AMI (EKS Optimized AMI). It supports on-demand or spot instances.
- **AWS Fargate** - no maintenance required, no nodes managed.

▼ ☑ **EKS Auto Scaling.**

EKS Metrics Server - lightweight, scalable aggregator of resource usage data in our Kubernetes cluster. The primary use of the Metrics Server is to enable Horizontal Pod Autoscaling (HPA). HPA automatically adjusts the number of pod replicas in a deployment, replication controller, or stateful set based on observed CPU utilization.

Kubernetes Cluster Autoscaler automatically adjusts the size of the Kubernetes cluster (e.g. the number of nodes) based on the resource requirements of the workloads running on the cluster.



Karpenter - flexible, high-performance Kubernetes cluster autoscaler that launches appropriately sized compute resources, like Amazon EC2 instances, in response to changing application load. It integrates with AWS to provision compute resources that precisely match workload requirements.

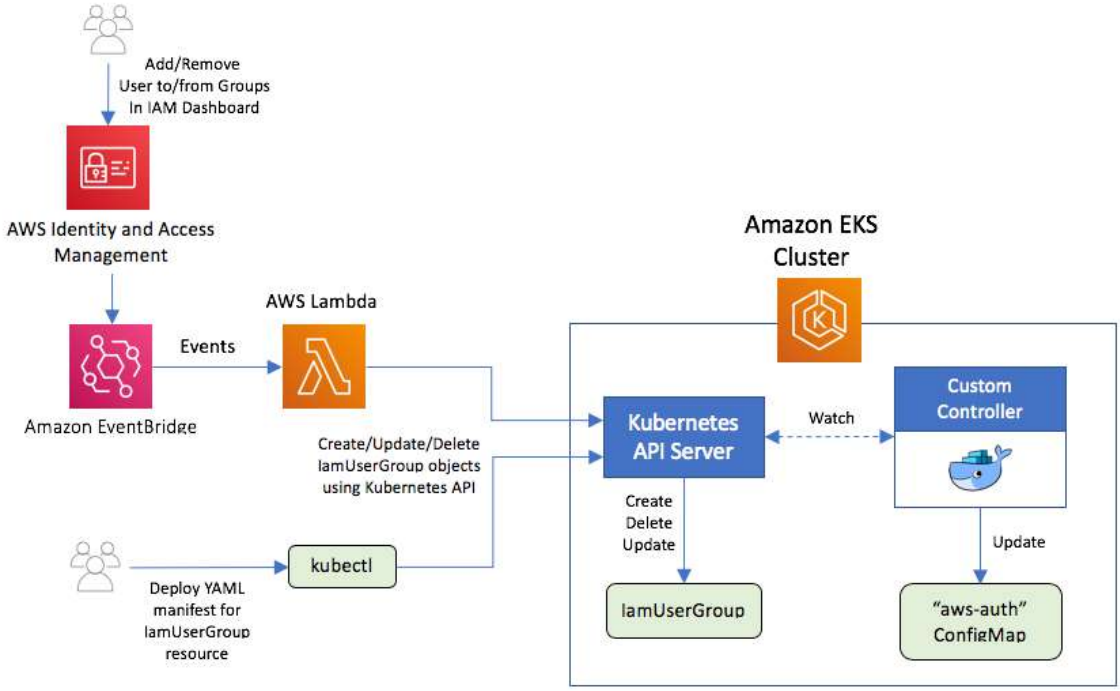
▼ ☑ **EKS Security.**

EKS uses IAM to provide authentication to our Kubernetes cluster, but it still relies on native Kubernetes Role-Based Access Control (RBAC) for authorization. This means that IAM is only used for the authentication of valid IAM entities. All permissions for interacting with our Amazon EKS cluster's Kubernetes API are managed through the native Kubernetes RBAC system.

Access to our cluster using IAM entities is enabled by the IAM Authenticator for Kubernetes, which runs on the EKS control plane. The authenticator gets its

configuration information from the *aws-auth ConfigMap* (AWS authenticator configuration map).

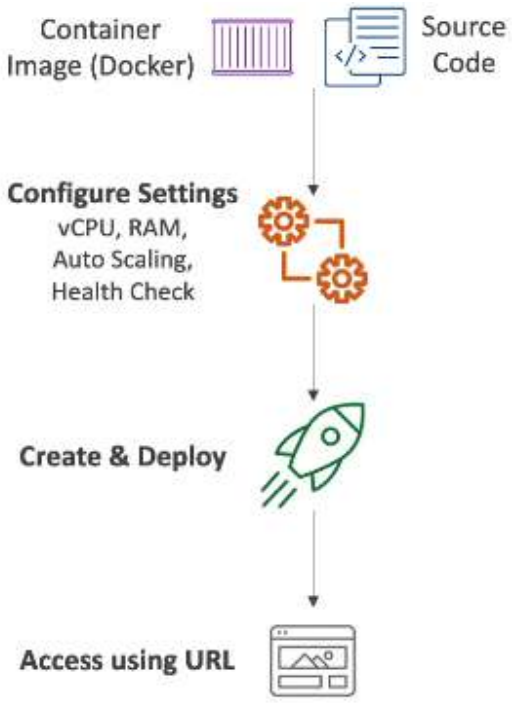
The **aws-auth ConfigMap** is automatically created and applied to our cluster when we create a managed node group or when we create a node group using **eksctl**. It is initially created to allow nodes to join our cluster, but we also use this ConfigMap to add role-based access control (RBAC) access to IAM users and roles.



▼ ☑ AWS App Runner.

App Runner is a fully managed service that makes it easy to deploy web applications and APIs at scale. It automatically builds and deploys the web app. We dont need intrastructure experience (start with our source code or container image).

We can connect to database, cache and message queue services. It has automatic scaling, load balancer and encryption.



Use cases: web apps, APIs, microservices, rapid production deployments.

AWS Serverless

▼ ☑ **AWS Lambda.**

AWS Lambda lets us run virtual functions on-demand (serverless). It has short execution and automated scaling. Functions get invoked by AWS only when needed (Event-Driven). Lambda functions are **stateless**, meaning each invocation is independent of previous invocations. State can be managed via external storage like DynamoDB or S3.

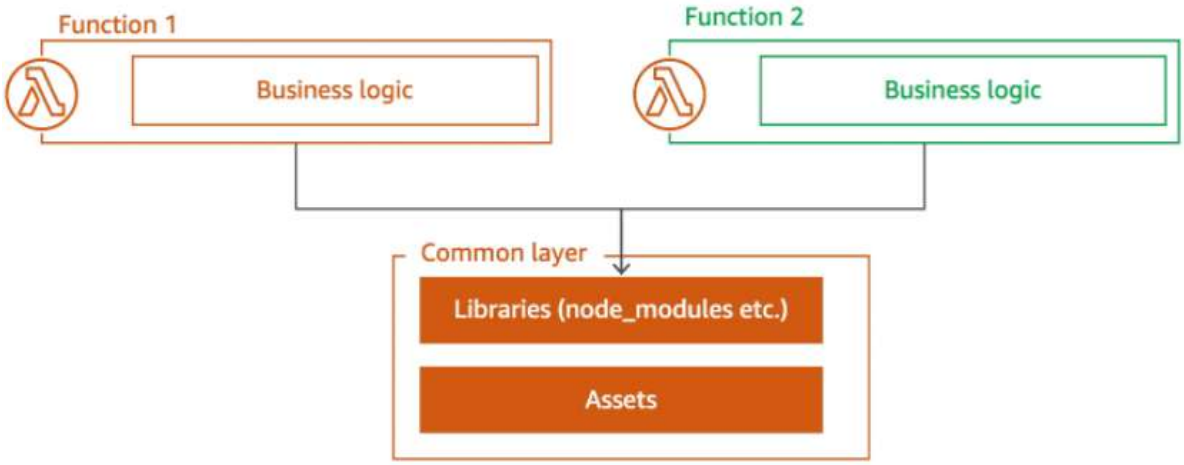
It is is integrated with the whole AWS suite of services and many programming languages. We can easily monitor AWS Lambda through AWS CloudWatch.

We **pay per request and compute time**. 1 million requests per month and 400 000 GB-seconds of compute time per month are free.

Lambda can be used for serverless CRON job (trigger an event at specific time).

Docker images can be run using Lambda. The container image must implement the Lambda Runtime API. We cant run Windows containers (only Linux).

AWS Lambda Layers - we can configure our AWS Lambda function to pull in additional code and content in the form of layers. A layer is a ZIP archive that contains libraries, a custom runtime, or other dependencies. With layers, we can use libraries in our function without needing to include them in your deployment package.



AWS Lambda URLs are a feature that allows us to create HTTP endpoints for our Lambda functions. This makes it easier to build serverless applications that interact with web clients or other HTTP-based services.

Lambda Limits

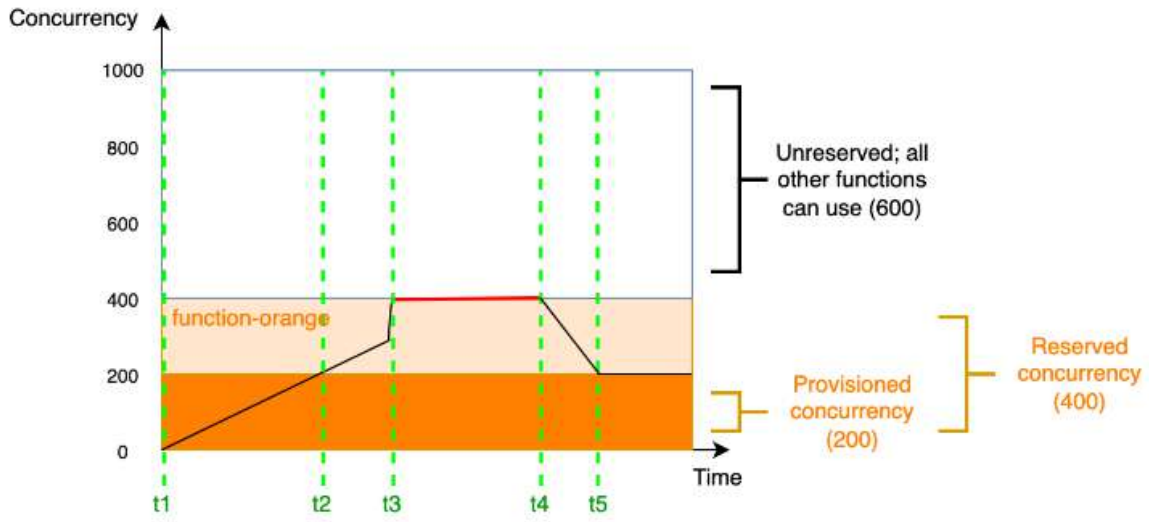
- Execution:
  - Memory allocation: 128 MB – 10GB (1 MB increments)
  - Maximum execution time: 900 seconds (15 minutes)
  - Environment variables (4 KB)
  - Disk capacity in the "function container" (in /tmp): 512 MB to 10GB
  - Concurrency executions: 1000 (can be increased)
- Deployment:
  - Lambda function deployment size (compressed .zip): 50 MB
  - Size of uncompressed deployment (code + dependencies): 250 MB
  - Can use the /tmp directory to load other files at startup
  - Size of environment variables: 4 KB

Since AWS Lambda functions can scale extremely quickly, it's a good idea to deploy a Amazon CloudWatch Alarm that notifies our team when function metrics such as ConcurrentExecutions or Invocations exceeds the expected threshold.

▼ ☑ AWS Lambda Scaling.

We can configure number of Lambda function which will execute simultaneously. Lambda automatically scales our function by running multiple instances of our code in parallel to handle high volumes of requests.

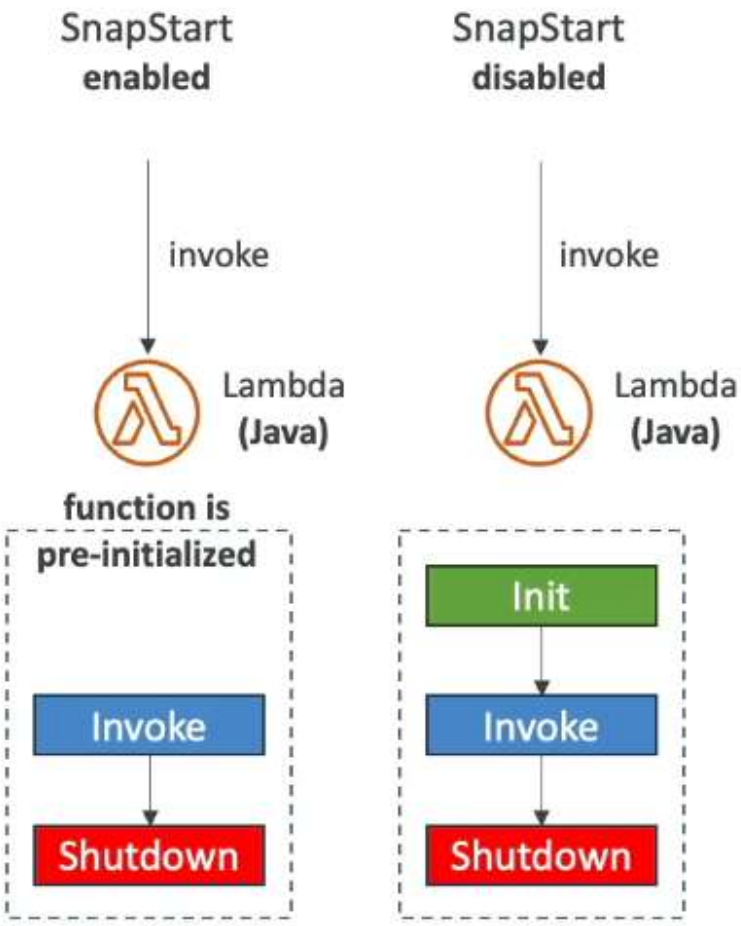
- **Provisioned Concurrency** - specified number of function instances are **pre-warmed** and ready to handle requests, which helps reduce latency and avoid cold starts. It's useful for functions with predictable traffic patterns or **latency-sensitive** applications.
- **Reserved Concurrency** - allows us to allocate a specific number of concurrent executions for a Lambda function, which ensures that the function always has the specified number of concurrent executions available and controls its maximum concurrency.



▼ ☑ AWS Lambda SnapStart.

Lambda SnapStart improves our Lambda functions performance up to 10x at **no extra cost** for Java 11 and above. When enabled, function is invoked from a preinitialized state (no function initialization from scratch).

When we publish a new version, Lambda initializes our function, takes a snapshot of memory and disk state of the initialized function and then snapshot is cached for low-latency access.



Lambda Invocation Lifecycle Phases

▼ ☐ **Edge Functions.**

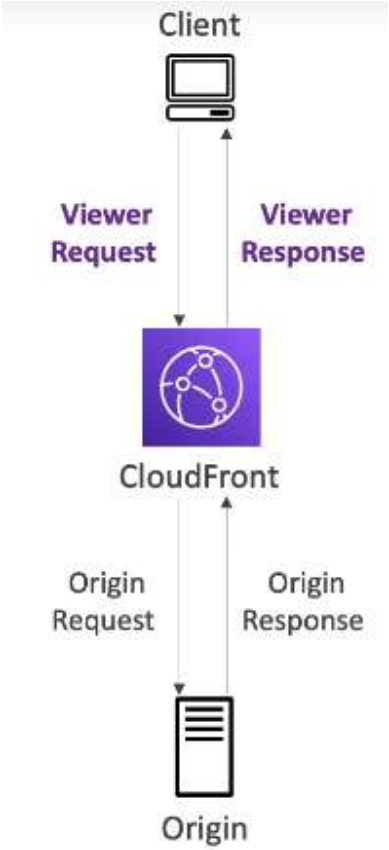
Edge Function - a code that we write and attach to CloudFront distributions. It runs close to our users to minimize latency. CloudFront provides two types: **CloudFront Functions** & **Lambda@Edge**. We dont have to manage any servers (fully serverles, pay only for what we use) and it is deployed globally.

Use cases: website security, dynamic web app at the edge, SEO (Search Engine Optimization), bot mitigation at the edge, real-time image transformation ...

CloudFront Functions - lightweight functions written in JavaScript. Used for high-scale, latency-sensitive CDN customizations (sub-ms startup times, millions of requests/second).

They are used to change viewer requests and responses.

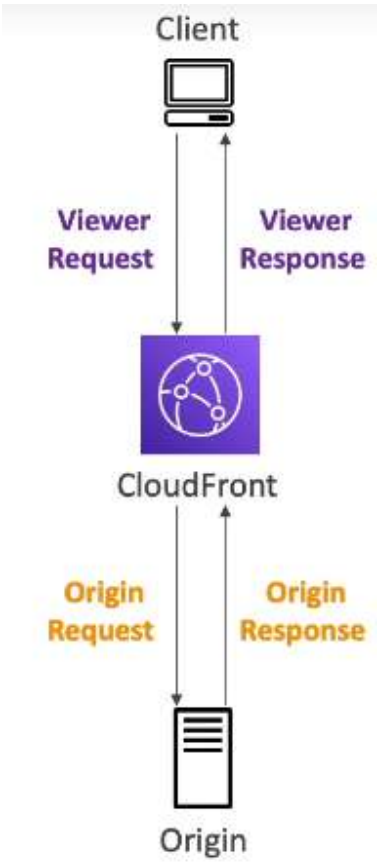
- Viewer Request - after CloudFront receives a request from a viewer.
- Viewer Response - before CloudFront forwards the response to the viewer.



Lambda@Edge - Lambda functions written in NodeJS or Python. It scales to 1000s of requests/seconds. They are used to change CloudFront requests and responses.

- Origin Request - before CloudFront forwards the request to the origin.
- Origin Response - after CloudFront receives the response from the origin.

Author our functions in one AWS Region (us-east-1), then CloudFront replicates to its locations.



CloudFront Functions vs Lambda@Edge

	CloudFront Functions	Lambda@Edge
Runtime Support	JavaScript	Node.js, Python
# of Requests	**Millions** of requests per second	**Thousands** of requests per second
CloudFront Triggers	- Viewer Request/Response	- Viewer Request/Response - Origin Request/Response
Max. Execution Time	< 1 ms	5 – 10 seconds
Max. Memory	2 MB	128 MB up to 10 GB
Total Package Size	10 KB	1 MB – 50 MB
Network Access, File System Access	No	Yes
Access to the Request Body	No	Yes
Pricing	Free tier available, 1/6th price of @Edge	No free tier, charged per request & duration

CloudFront Functions

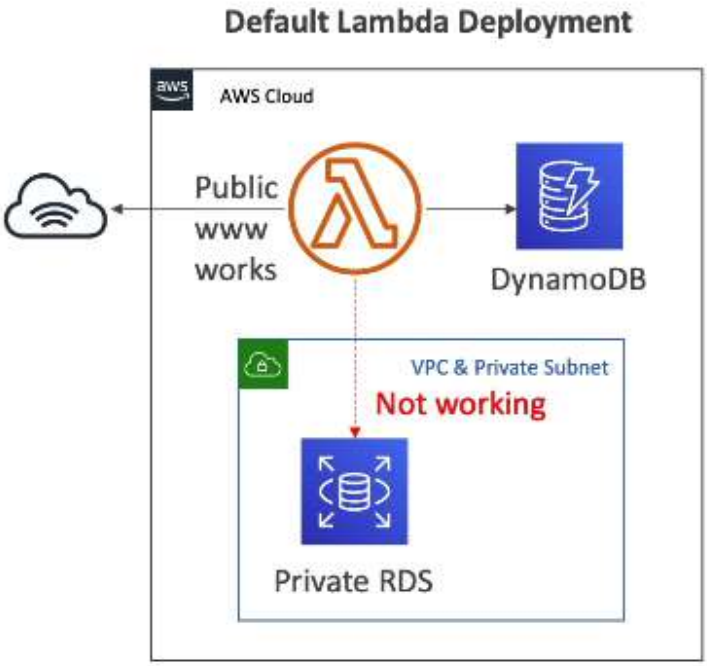
- Cache key normalization
  - Transform request attributes (headers, cookies, query strings, URL) to create an optimal Cache Key
- Header manipulation
  - Insert/modify/delete HTTP headers in the request or response
- URL rewrites or redirects
- Request authentication & authorization
  - Create and validate user-generated tokens (e.g., JWT) to allow/deny requests

Lambda@Edge

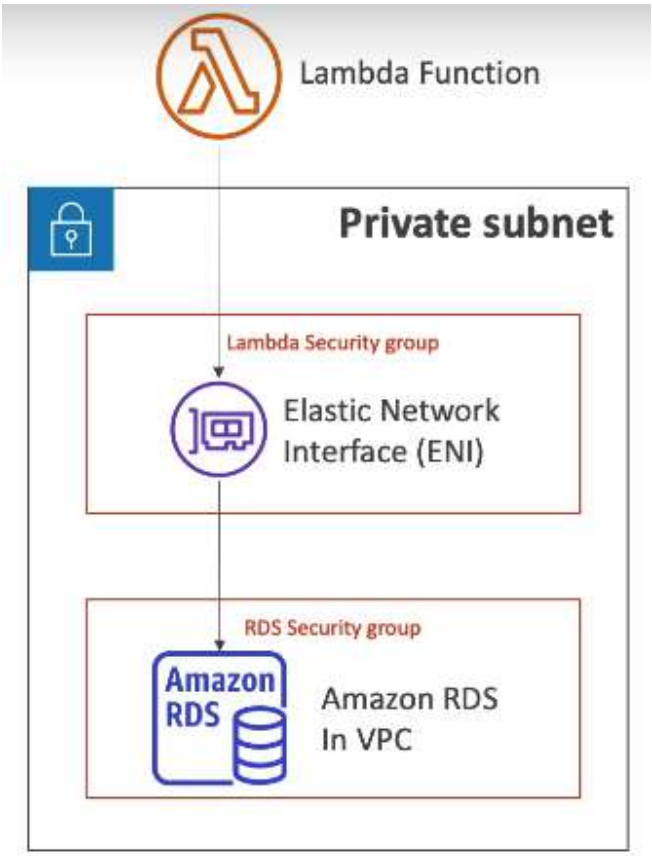
- Longer execution time (several ms)
- Adjustable CPU or memory
- Your code depends on a 3rd libraries (e.g., AWS SDK to access other AWS services)
- Network access to use external services for processing
- File system access or access to the body of HTTP requests

▼ ☑ **Lambda in VPC.**

By default, our Lambda function is launched outside our own VPC. Therefore, it cannot access resources in our VPC (RDS, ElastiCache, etc).



We must define the VPC ID, the subnets and security groups. Lambda will create an ENI in our subnets.

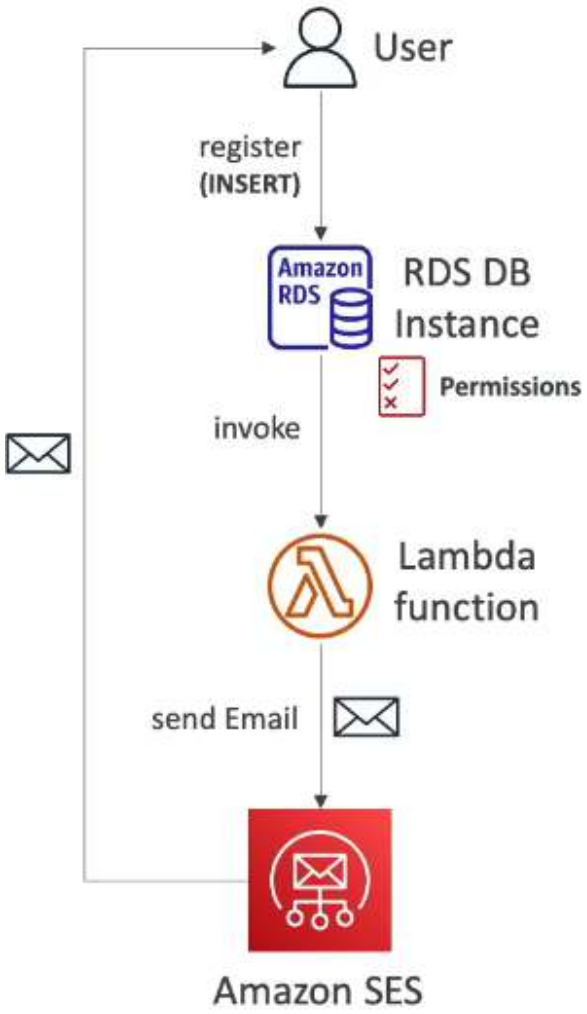


If Lambda functions directly access our database, they may open too many connections under high load. We can use RDS Proxy to improve scalability. The Lambda function must be deployed in our VPC, because RDS Proxy is never publicly accessible.

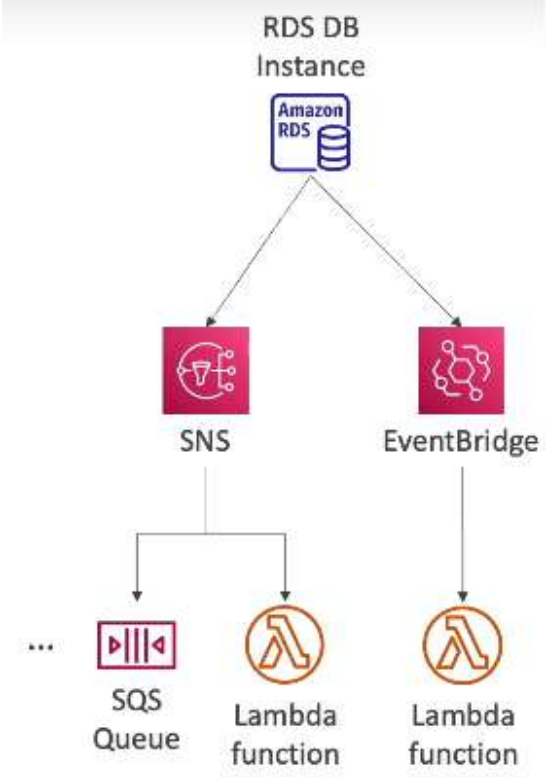
▼ ☐ **RDS - Invoking Lambda & Event Notifications.**

We can invoke Lambda functions from within our DB instance. It allows us to process data events from within a database. It's **supported for RDS for Postrgres and Aurora MySQL**.

We must allow outbound traffic to our Lambda function from within our DB instance (Public, NAT Gateway, VPC Endpoints). DB instance must have the required permissions to invoke the Lambda.



Event Notifications - notifications that tell information about the DB instance itself. It is near real-time event. We dont have any information about the data itself. Can send notifications to SNS or subscribe to events using EventBridge.

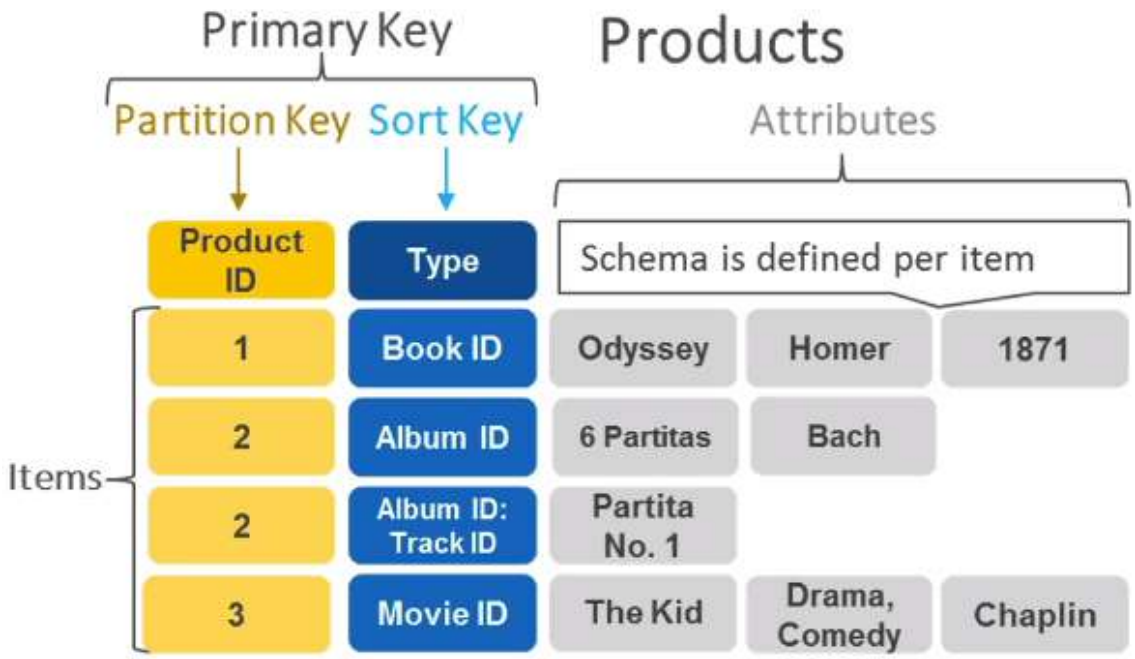


Need to subscribe to the following event categories: DB instance, DB snapshost, DB Parameter Group, DB Security Group, RDS Proxy, Custom Engine Version.

▼ ☑ **DynamoDB.**

DynamoDB is fully managed and highly available NoSQL schemaless database with replication across 3 AZ. It's distributed serverless database which can scale to massive workloads. DynamoDB is fast and consisent in performance (can handle millions of requests per second, trillions of row and 100s of TB of storage). It's low cost and has auto scaling capabilities. It can rapidly evolve schemas. It has **single-digit millisecond latency** (low latency retrieval).

DynamoDB is integrated with IAM for security, authorization and administration.



DynamoDB uses the **partition key** value as input to an internal hash function. The output from the hash function determines the partition in which the item will be stored. All items with the same partition key are stored together, in sorted order by sort key value. If no sort key is used, no two items can have the same partition key value.

Read/Write Capacity Modes

- **Provisioned Mode** (default) - we specify the number of reads/writes per second and need to plan capacity beforehand. We pay for provisioned Read Capacity Units (RCU) & Write Capacity Units (WCU). It is possible to add auto scaling mode for RCU & WCU.

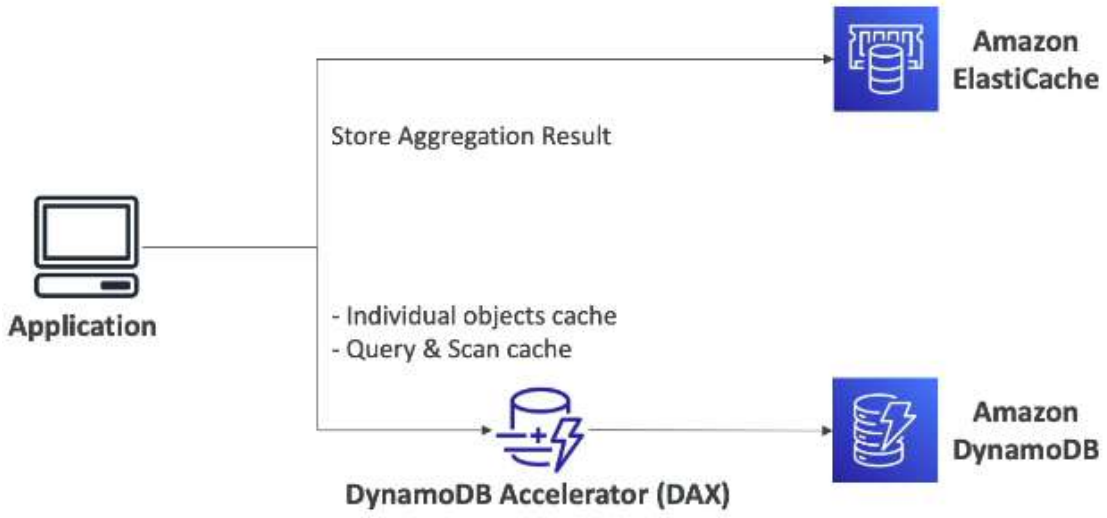
  The more distinct partition key values that our workload accesses, the more those requests will be spread across the partitioned space.

- **On-demand Mode** - read/writes automatically scale up/down with our workloads. No capacity planning is needed. We pay for what we use (more expensive). It is great for unpredictable workloads, steep sudden spikes.

▼ ☑ **DynamoDB Advanced Features.**

DynamoDB Accelerator (DAX) - fully managed, secure, highly scalable and highly available in-memory cache for DynamoDB. It gives 10x performance

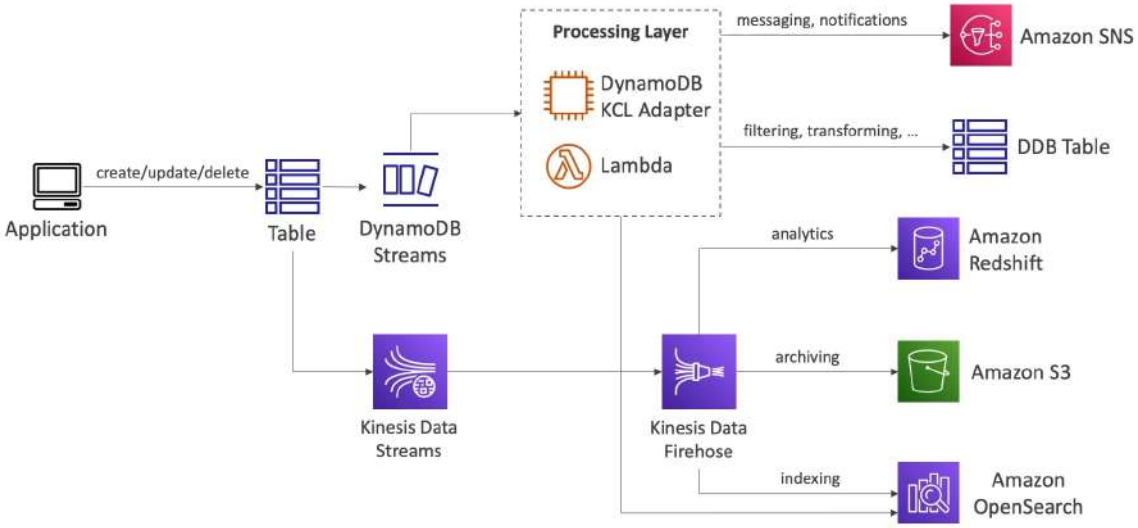
improvement (single digit millisecond latency to microseconds latency). It doesnt require application logic modification (compatible with existing DynamoDB APIs). It has 5 minutes TTL for cache (default).



DynamoDB Stream Processing - ordered stream of item-level modifications (create/update/delete) in a table.

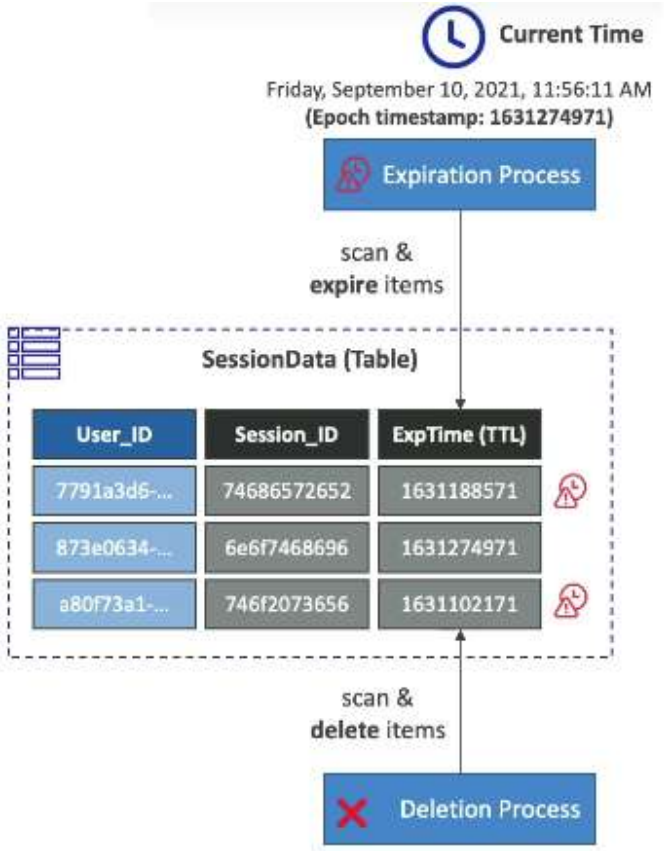
It is used for reacting to changes in real time, real time usage analytics, implementing cross-region replication, invoke AWS Lambda on changes to our DynamoDB table...

- **DynamoDB Streams** - process using AWS Lambda Triggers or DynamoDB Stream Kinesis adapter. We have 24h retention and it has limited number of consumers.
- **KDS** - Process using AWS Lambda, KDA, KDF, AWS Glue Streaming ETL... We have 1 year retention and it has high number of consumers.



DynamoDB Global Tables - feature that makes DynamoDB table accessible with low latency in multiple regions. With this we can read and write to any AWS Region and it is called Active-Active Replication. We must enable DynamoDB Streams as a pre-requisite.

DynamoDB TTL - automatically delete items after an expiry timestamp. Used to reduce stored data by keeping only current items, adhere to regulatory obligations, web session handling...



DynamoDB Backups

- **Continuous backups using PITR** - optionally enabled for the last 35 days. PITR to any time within the backup window. The recovery process creates a new table.
- **On-demand backups** - full backups for long-term retention, until explicitely delted. It doesnt affect performance or latency. Can be configured and managed in AWS Backup. The recovery process creates a new table.

DynamoDB Integration with S3

- **Export to S3** (**must enable PITR**) - works for any point of time in the last 35 days. It doesnt affect the read capacity of our table. We can perform data analysis on top of DynamoDB and ETL on top of S3 data before importing back to DynamoDB.
- **Import to S3** - import CSV, DynamoDB JSON or ION format. It doesnt consume any write capacity. It creates a new table. Import errors are logged in CloudWatch Logs.

▼ ☑ API Gateway.

Amazon API Gateway is a fully managed service for developers to easily create, publish, maintain, monitor and secure APIs. It is serverless and scalable, supports RESTful APIs and WebSocket APIs. Has support for security, user authentication, API keys... We can validate requests and responses (also cache API responses). We pay only for the API calls we receive and the amount of data transferred out.

We can monitor API usage, latency, and error rates using CloudWatch.

API Gateway Integrations

- **Lambda Function** - can invoke AWS Lambda functions within our account and start AWS Step Functions state machines.
- **HTTP** - can make HTTP requests to endpoints hosted on AWS services such as Elastic Beanstalk and EC2, as well as to non-AWS HTTP endpoints accessible over the public internet.
- **AWS Services** - can be directly integrated with other AWS services. For example, we can configure an API method to send data directly to Amazon Kinesis.

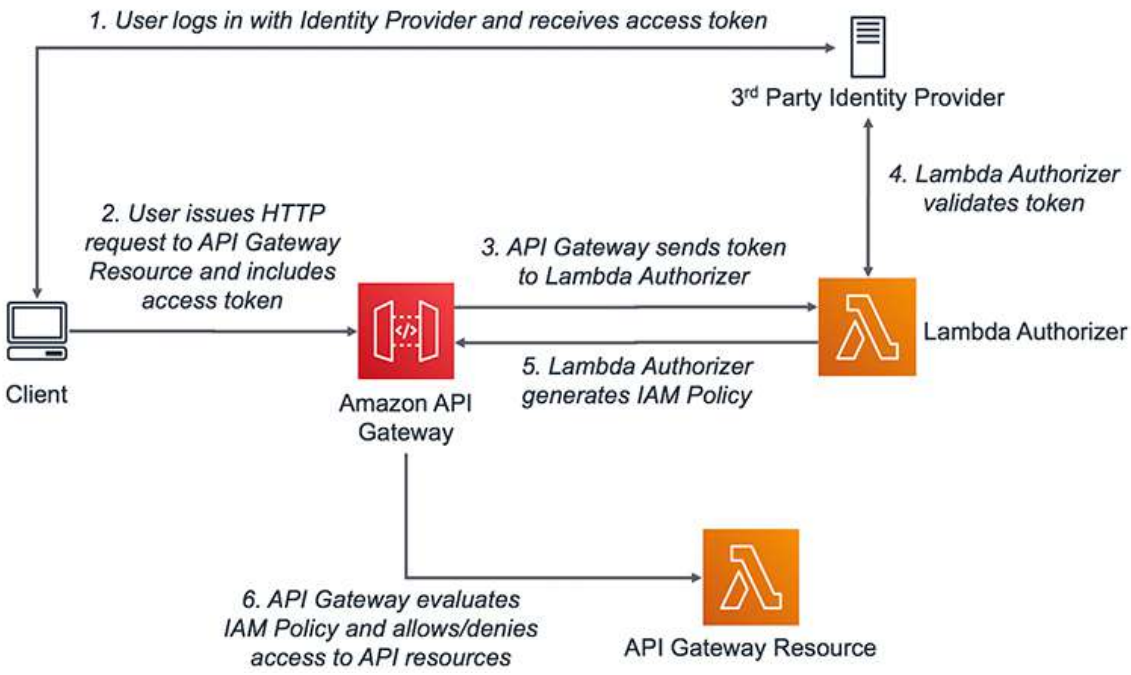
API Gateway Endpoint Types

- **Edge-Optimized** (default) - for global clients. Requests are routed through the CloudFront Edge locations. The API Gateway still lives in only one region.
- **Regional** - for clients within the same region. Cloud manually combine with CloudFront (more control over the caching strategies and the distribution).
- **Private** - can only be accessed from our VPC using an interface VPC endpoint (ENI). We need to use a resource policy to define access.

API Gateway Security

User authentication through IAM roles (for internal apps), cognito (for external users) or custom authorizer (our own logic).

Lambda Authorizer (Custom Authorizer) is a feature of AWS API Gateway that allows us to control access to our APIs using custom authentication and authorization logic implemented in AWS Lambda functions.



Custom Domain Name HTTPS security through integration with ACM. If using Edge-Optimized endpoint, then the certificate must be in us-east-1. If using Regional endpoint, the certificate must be in the API Gateway region. We must setup CNAME or Alias record in Route 53.

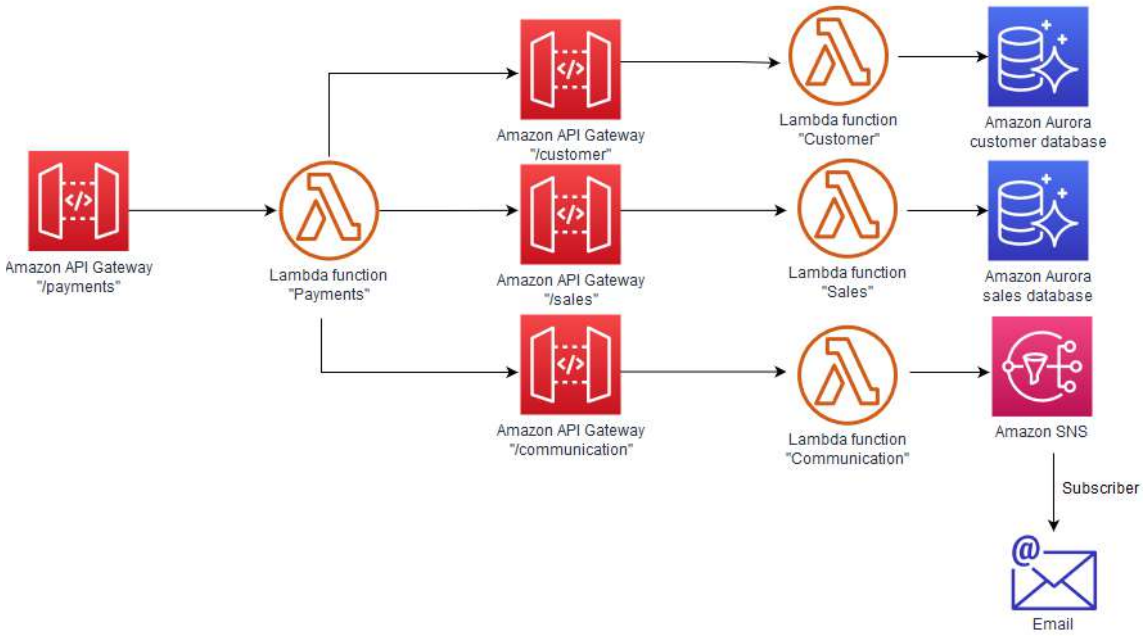
▼ ☑ **API Gateway Advanced Features.**

API Usage Plan defines who can access our API, how much they can access, and how the usage is monitored and throttled. It helps us manage and control API traffic, set quotas, and monitor usage metrics.

Custom Domain Names allows us to use our own domain names for our APIs, providing a more professional and user-friendly experience. We can map multiple APIs to different paths on a single domain using base path mappings.

▼ **Solution Architecture - Serverless Microservice Architecture.**

We can break down applications into smaller, loosely coupled services that are independently deployable and scalable without managing servers. It scales automatically and we only pay for what you use.

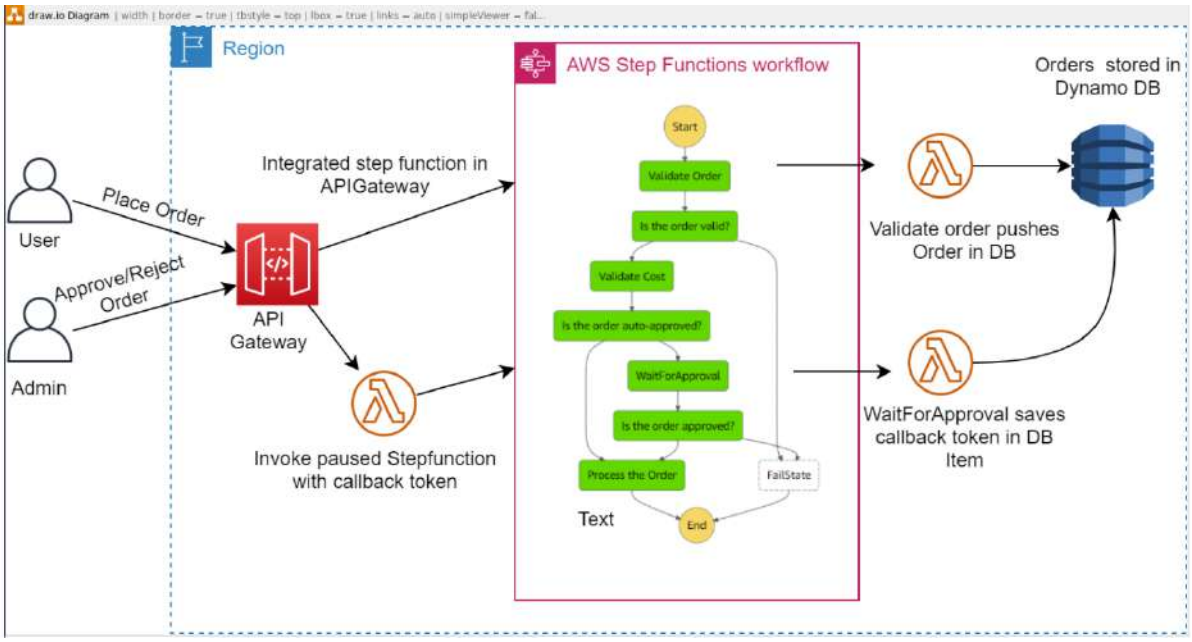


▼ ☑ AWS Step Functions.

Step Functions is a serverless orchestration service that allows us to coordinate multiple AWS services into serverless workflows. It orchtestrates our Lambda functions.

It can be integrated with EC2, ECS, on-remises servers, API Gateway, SQS, etc ...

We can use it for implementing human approval feature.



Step Functions Local is a tool that allows us to run and test our Step Functions state machines locally on our development machine. It can help us develop and debug your workflows without needing to deploy them to the AWS cloud, which can save time and costs.

☐ AWS Step Functions Map State

AWS Step Functions' Map State is a feature designed to iterate over a collection of items and process each one in parallel or sequentially, depending on how we configure it. It allows us to specify a JSON array (or list) as input and then iterate over that array and apply a specified state machine (or sub-state machine) to each item.

The Map state allows us to configure how many iterations are processed in parallel. We can set a MaxConcurrency value to limit the number of parallel executions.

- **Inline Mode** - each item in the array is processed using the same state machine specified directly within the Map state's Iterator. State machine logic defined in the Iterator is directly applied to each item in the array.

```
{
  "Type": "Map",
  "ItemsPath": "$.items",
  "Iterator": {
    "StartAt": "ProcessItem",
    "States": {
      "ProcessItem": {
        "Type": "Task",
        "Resource": "arn:aws:lambda:REGION:ID:function:Proc
        "End": true
      }
    }
  },
  "End": true
}
```

- **Distributed Mode** - the Map state is used in conjunction with a nested state machine or sub-state machine that is applied to each item in the array. It

allows a more complex or different workflow to be used for each item. It is designed specifically for **massive parallelism** by breaking up large-scale processing tasks into smaller batches.

```
{
  "Type": "Map",
  "ItemsPath": "$.items",
  "Iterator": {
    "StartAt": "SubMachine",
    "States": {
      "SubStateMachine": {
        "Type": "StateMachine",
        "StateMachineArn": "arn:aws:states:REGION:ID:stateM
        "End": true
      }
    }
  },
  "End": true
}
```

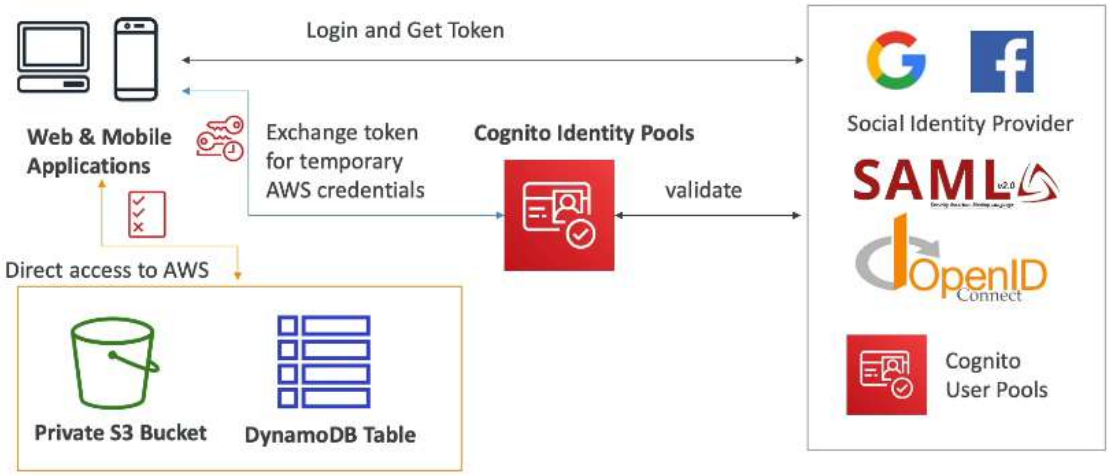
▼ ☑ **Amazon Cognito.**

Cognito gives users an identity to interact with our web or mobile applications.

- **Cognito User Pools (CUP)** - serverless database of user for our web & mobile apps. It provides sign in functionality for app users. Integration with **API Gateway** & **ALB**. It provides simple login (username/mail and password combination), password reset, 2FA, MFA...

- **Cognito Identity Pools (Federated Identity)** - used to get identities for "users" so they obtain temporary AWS credentials. Users source can be CUP, third party logins, etc...

  Users can then access AWS services directly or through API Gateway. The IAM policies applied to the credentials are defined in Cognito. There are default IAM roles for authenticated and guest users.



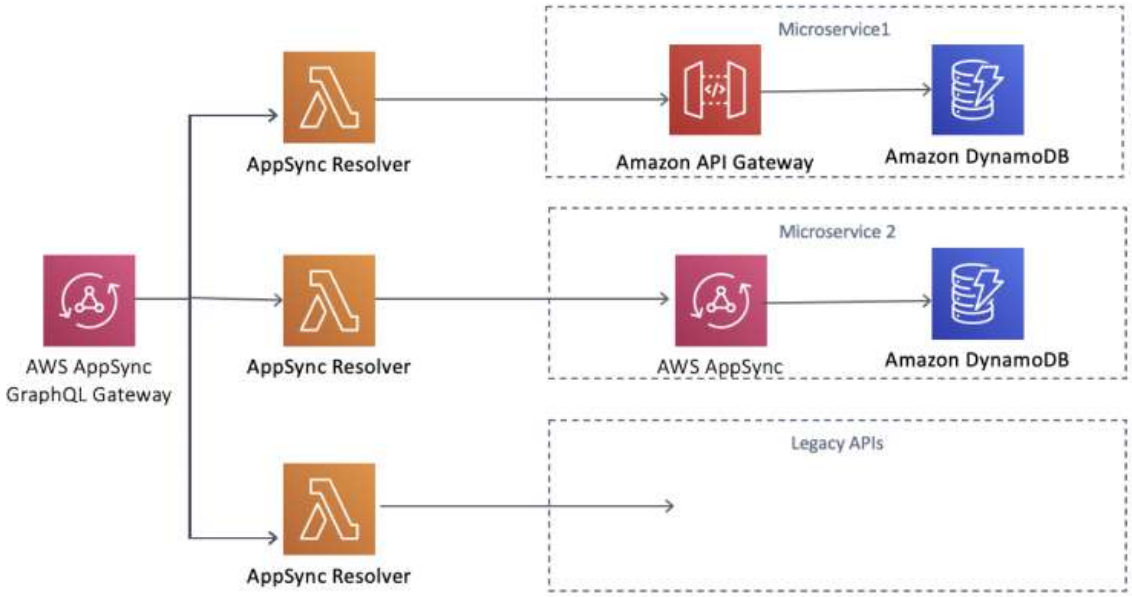
Row Level Security in DynamoDB

▼ ☐ **AWS AppSync.**

AWS AppSync is a fully managed service that simplifies the development of GraphQL and real-time APIs by enabling us to build scalable applications with ease. It **acts as a bridge** between our clients (such as web or mobile apps) and our data sources (like DynamoDB, Lambda, RDS, and other AWS services). It manages the complex backend work such as authentication, **caching**, data synchronization, and access control.

AppSync provides real-time updates to connected clients, making it ideal for use cases requiring instant data synchronization.

- **AppSync Resolvers** - determine how GraphQL queries or mutations should fetch or manipulate data.
- **AppSync Mapping Templates** - transform incoming requests into the appropriate format for the backend and then return the response.

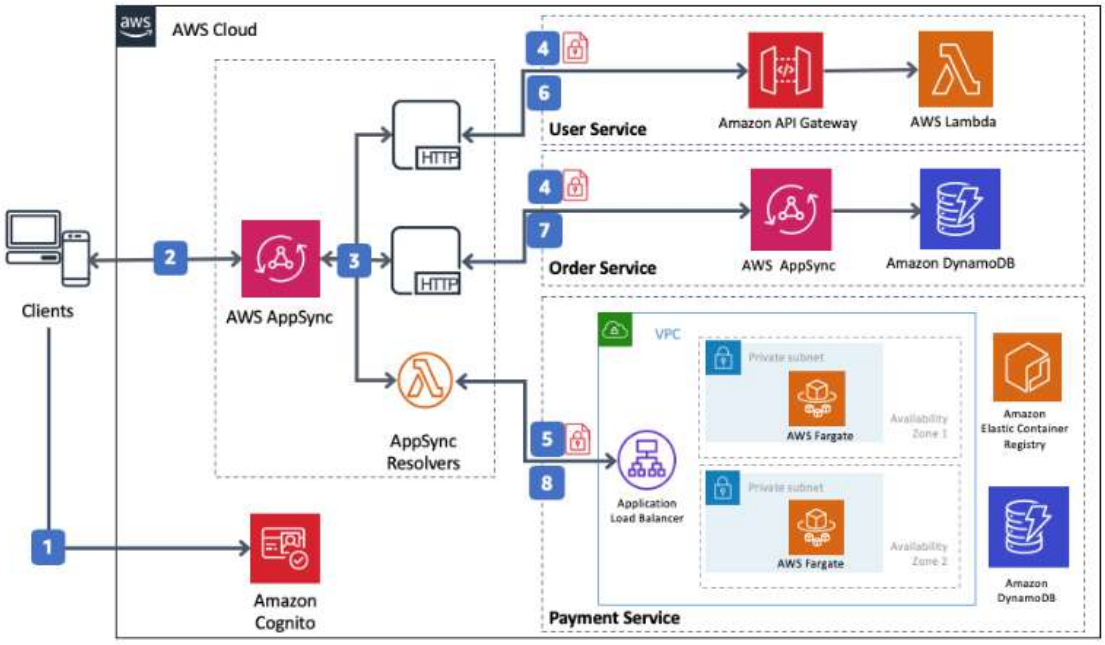


AppSync Security

AppSync integrates with multiple AWS authentication providers, such as **Amazon Cognito**, API keys, IAM, or OpenID Connect (OIDC). We can implement fine-grained access control using role-based permissions and authorization rules.

Field-level Authorization - we can restrict access down to specific fields in our GraphQL schema, ensuring only authorized users can access sensitive data.

▼ **Solution Architecture - Access to Multiple Microservices.**



Section III

Databases in AWS

▼ ☑ **RDS Summary.**

- Managed Postgres/MySQL/Oracle/SQL Server/DB2/MariaDB/Custom.
- Provisioned RDS instance size & EBS Volume type & size.
- Auto scaling capability for storage.
- Support for Read Replicas & Multi-AZ.
- Security through IAM, Security Groups, KMS, SSL in transit.
- Automated backup with PITR (35 days).
- Manual DB Snapshots for longer-term recovery.
- Managed and scheduled maintenance (with downtime).
- Support for IAM authentication, integration with Secrets Manager.
- RDS Custom for access to and customize the underlying instance (Oracle & SQL Server).
- Used for RDBMS & OLTP.

▼ ☑ **Aurora Summary.**

- Compatible API for Postgres & MySQL, separation of storage and compute.
- Data is stored in 6 replicas, across 3 AZ - highly available, self-healing, auto scaling.
- Cluster of DB instance across multiple AZ, auto scaling of Read Replicas.
- Custom endpoints for writer and reader DB instances.
- Same security/monitoring/maintenance features as RDS.
- Aurora Serverless - unpredictable/intermittent workloads, no capacity planning.
- Aurora Global - up to 16 DB Read Replicas in each region, < 1s storage replication.
- Aurora ML - perform ML using SageMaker & Comprehend on Aurora.
- Aurora Database Cloning - new cluster from existing one, faster than restoring a snapshot.
- Used for same as RDS, but with less maintenance/more flexibility/more performance/more features.

▼ ☑ **ElastiCache Summary.**

- Managed Redis/Memcached (similar offering as RDS, but for caches).
- In-memory data store, sub-millisecond latency.
- Select an ElastiCache instance type (e.g, cache.m6g.large).
- Support for Clustering (Redis) and Multi AZ, Read Replicas (sharding).
- Security through IAM, Security Groups, KMS, Redis Auth.
- Backup/Snapshot/PITR feature.
- Managed and scheduled maintenance.
- **Requires some application code changes to be leveraged**.
- Used for key/value store, frequent reads, less writes, cache results for DB queries, store session data for websites, cannot use SQL.

▼ ☑ **DynamoDB Summary.**

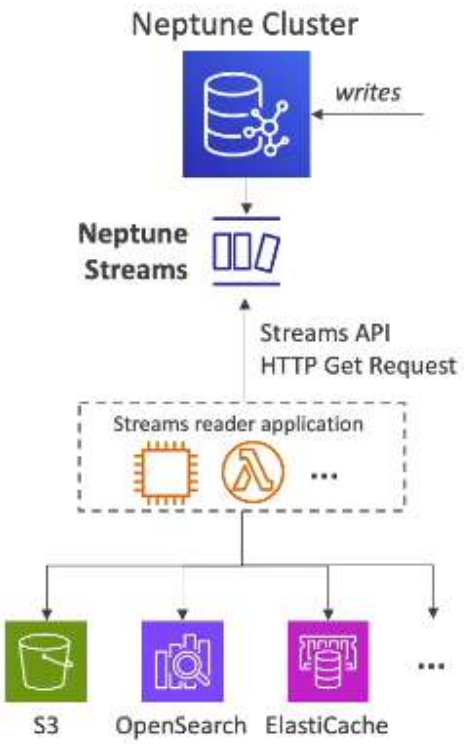
- AWS proprietary technology, managed serverless NoSQL database, millisecond latency.
- Capacity modes: provisioned capacity with optional auto scaling or on-demand capacity.
- Can replace ElastiCache as a key/value store (storing session data for example, using TTL).
- Highly available, Multi-AZ by default, read & writes are decoupled, transaction capability.
- DAX cluster for read cache, microsecond read latency.
- Security, authentication and authorization is done through IAM.
- Event processing: DynamoDB Streams to integrate with AWS Lambda or KDS.
- Global Table feature: active-active setup.
- Automate backups up to 36 days with PITR or on-demand backups.
- Export to S3 without using RCU within the PITR window, import from S3 without using WCU.
- Great to rapidly evolve schemas.
- Used for serverless applications development and distributed serverless cache.

▼ ☑ **Amazon Neptune.**

Neptune is a fully managed graph database. It is highly available across 3 AZ, with up to 15 Read Replicas. We can build and run apps working with highly connected datasets (optimized for these complex and hard queries). It can store up to billions of relations and query the graph with milliseconds latency.

Neptune Streams - real-time ordered sequence of every change to our graph data. Changes are available immediately after writing. There are no duplicates and order is strict. Streams data is accesible in an HTTP REST API.

Use cases: send notifications when certain changes are made, maintain our graph data synchronized in another data store, replicate data across regions in Neptune.



▼ ☑ **Amazon Keyspaces.**

Keyspaces is a managed Apache Cassandra-compatible database service. It is serverless, scalable, highly available, fully managed by AWS. Tables are replicated 3x across multiple AZ.

It uses CQL (Cassandra Query Language) and has single-digit millisecond latency at any scale. For capacity we have on-demand mode or provisioned mode with auto-scaling.

Use cases: store IoT devices info, time-series data, etc...

▼ ☑ **Amazon DocumentDB.**

DocumentDB is "AWS implementation" of MongoDB. It has similar deployment concepts as Aurora (fully managed, highly available with replication across 3 AZ). Its storage automatically grows in increments of 10GB.

Automatically scales to workloads with millions of requests per second.

▼ ☑ **Amazon QLDB & Managed Blockchain.**

QLDB (Quantum Ledger Database) is a fully managed, serverless, high available (across 3 AZ) ledger database that provides a transparent, **immutable** (no entry

can be removed or modified), and cryptographically verifiable transaction log. In QLDB there is no decentralization.

It's used to review history of all the changes made to our application data over time.

Has 2-3x better performance than common ledger blockchain frameworks.

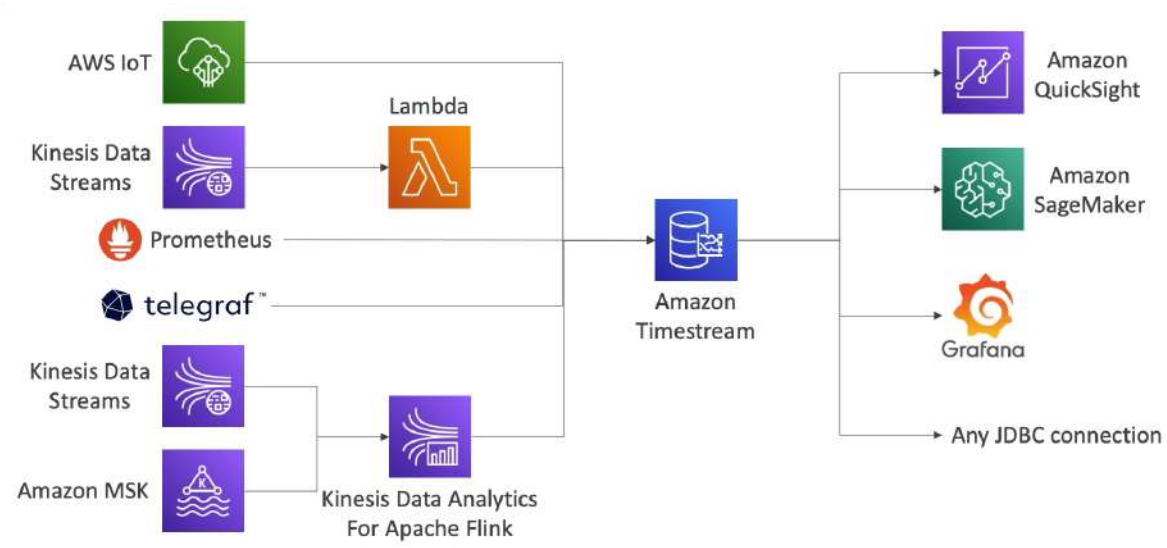
Managed Blockchain is a fully managed decentralized service that simplifies the creation and management of scalable blockchain networks. It is compatibile with Hyperledger Fabric & Ethereum.

▼ ☑ **Amazon Timestream.**

Timestream is a fully managed, fast, scalable serverless time series database. It autimatically scales up/down to adjust capacity. Can store and analyze trillions of events per day. It is 1000s times fasters & 1/10th the cost of relational databases.

It is compatible with SQL and we can use scheduled queries. Recent data is kept in memory and historical data kept in a cost-optimized storage. It has encryption in transit and at rest.

Use cases: IoT apps, operational apps, real-time analytics, etc...



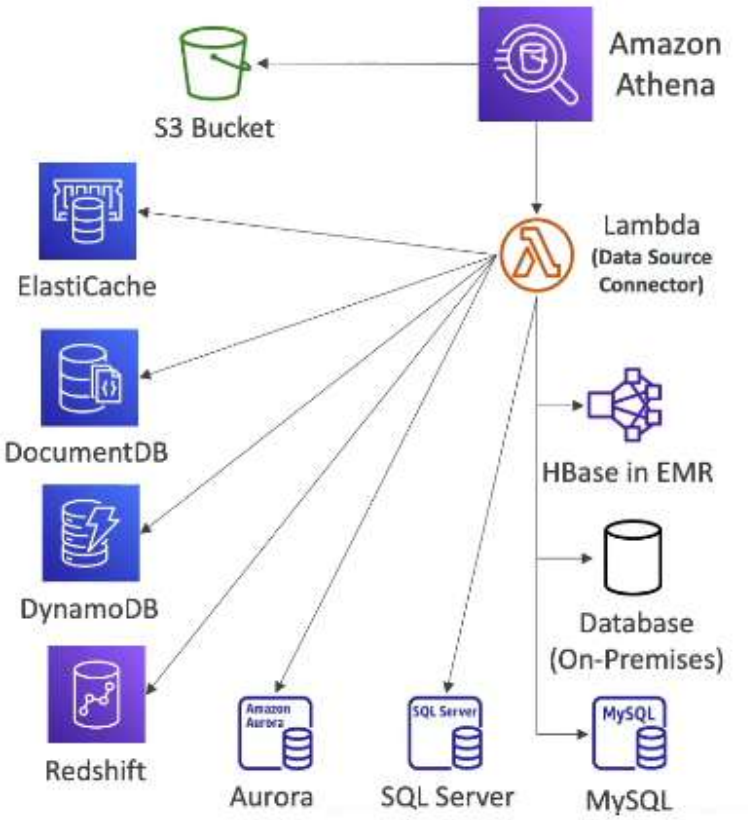
Data & Analytics

▼ ☑ **Amazon Athena.**

Athena is a serverless query service which is used to analyze data stored in S3. It uses standard SQL to query the files and supports CSV, JSON, ORC, Avro and Parquet. It is commonly used with Quicksight for reporting/dashboards.

Use cases: business intelligence, analytics, reporting, analyzing querying VPC Flow Logs, CloudTrail trails, etc...

Athena Federated Query - allows us to run SQL queries across data stored in relational, non-relational, object and custom data sources. It uses Data Source Connectors that run on AWS Lambda and it stores results back in S3.



Athena Performance Improvement

- Use columnar data for cost-savings (less scan) - Apache Parquet or ORC is recommended. We can use Glue to convert our data to Parquet or ORC.

Dataset	Size on Amazon S3	Query Run time	Data Scanned	Cost
Data stored as CSV files	1 TB	236 seconds	1.15 TB	$5.75
Data stored in Apache Parquet format*	130 GB	6.78 seconds	2.51 GB	$0.01
Savings / Speedup	87% less with Parquet	34x faster	99% less data scanned	99.7% savings

- Compress data - for smaller retrievals.
- Partition datasets in S3 - easy querying on virtual columns (*s3://athena-ex/flight/parquet/year=1995/month=1/day=1*).
- Use larger files (> 128MB) to minimize overhead.

▼ ☑ **Amazon Redshift.**

Redshift is based on Postgres, but it's not used for OLTP (Online Transaction Processing). It is used for **OLAP** (Online Analytical Processing - Analytics & Data

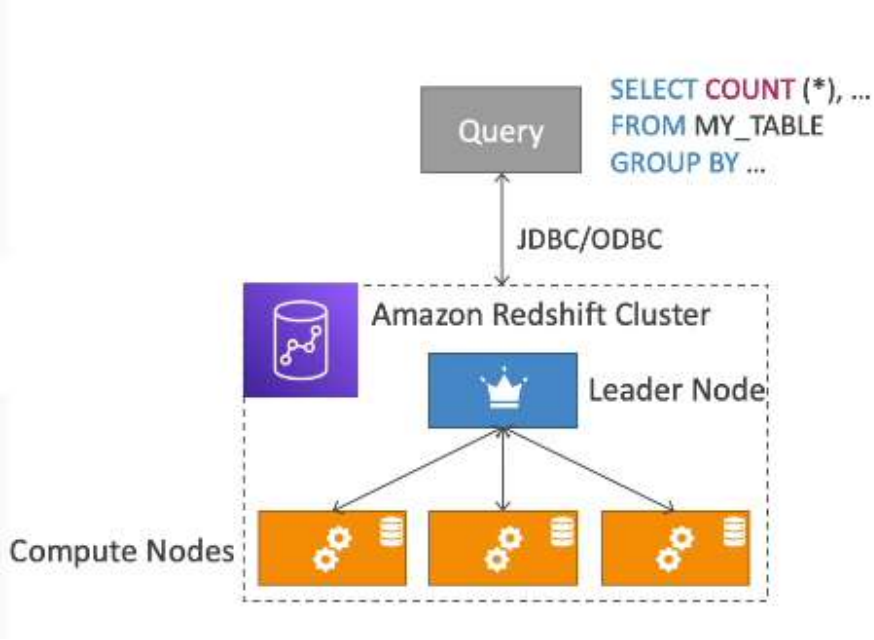
Warehousing). Redshift loads data once every hour, not every second and it has 10x better performance than other data warehouses, scale to PBs of data. Has Massively Parallel Processing (MPP), highly available.

Instead of row based storage of data Redshift uses columns. In comparison with Athena, Redshift has faster queries, joins and aggregations because it uses indexes.

Redshift has pay as you go type of pricing based on the instances provisioned. By leveraging **cross-region snapshot copying**, we can significantly enhance our Redshift cluster's resilience and ensure that our data is protected against regional disruptions.

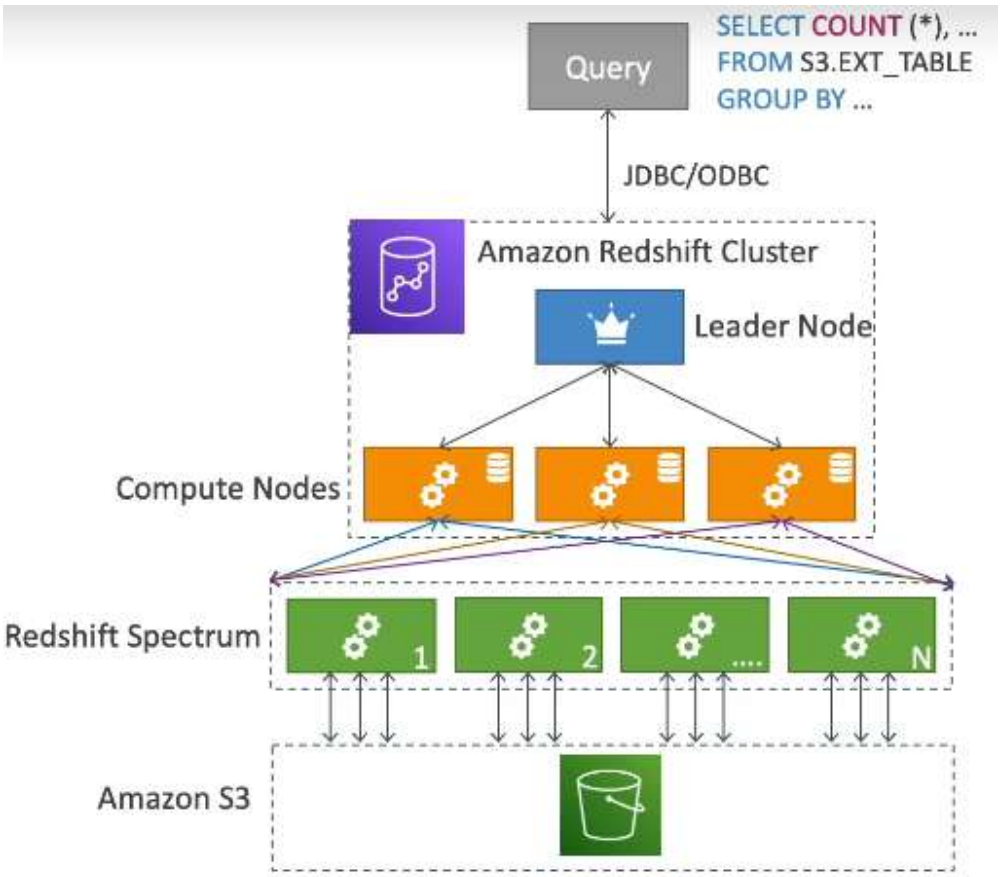
Redshift Cluster

- **Leader Node** - for query planning and results aggregation.
- **Compute Node** - for performing the queries and sending results to leader.



We must provision the node size in advance and use RI for cost savings.

Redshift Spectrum - feature which lets us query data that is already in S3 without loading it. For this we must have a Redshift cluster available to start the query.

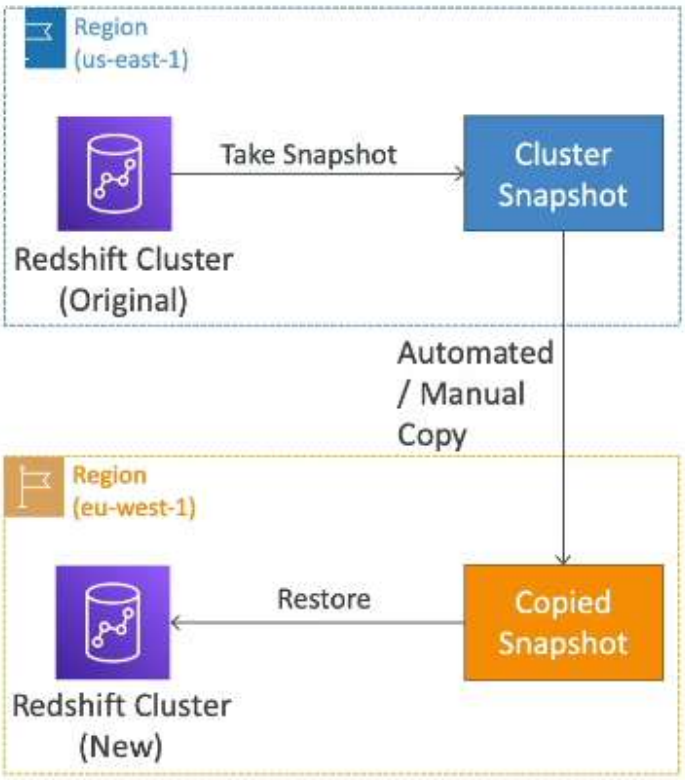


Redshift Snapshots & DR

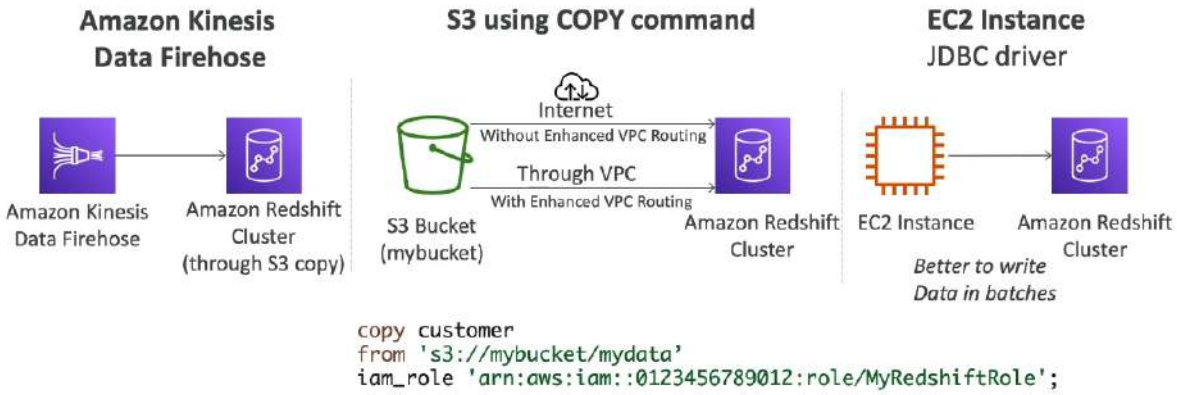
Redshift has Multi-AZ mode for some clusters. Snapshots are PIT backups of a cluster, stored internally in S3. They are incremental (only what has changed is saved). We can restore a snapshot into a new cluster.

- **Automated snapshots** - every 8h, every 5GB or on a schedule. Need to set retention period.
- **Manual snapshots** - snapshot is retained until we delete it.

We can configure Redshift to automatically copy snapshots of a cluster to another AWS Region.



Loading Data into Redshift

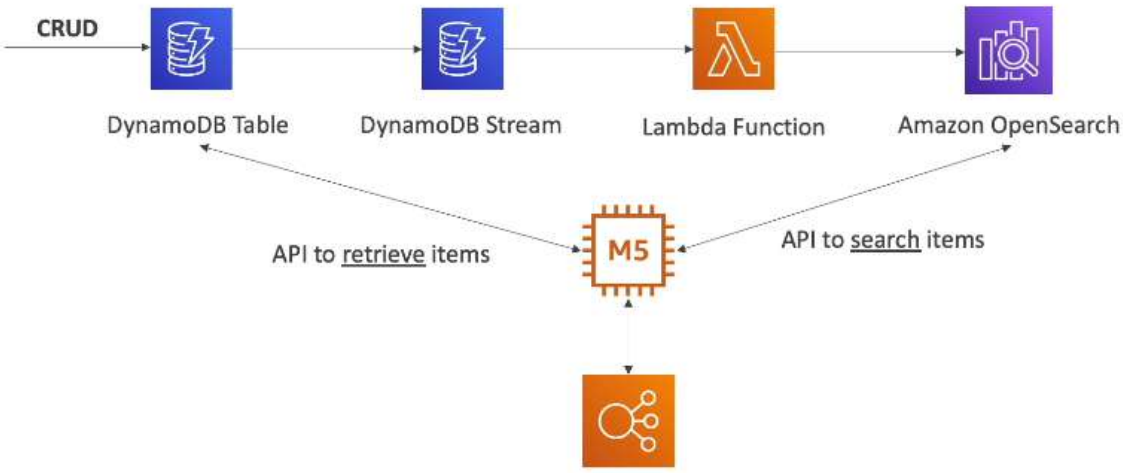


▼ ☑ **Amazon OpenSearch (ElasticSearch).**

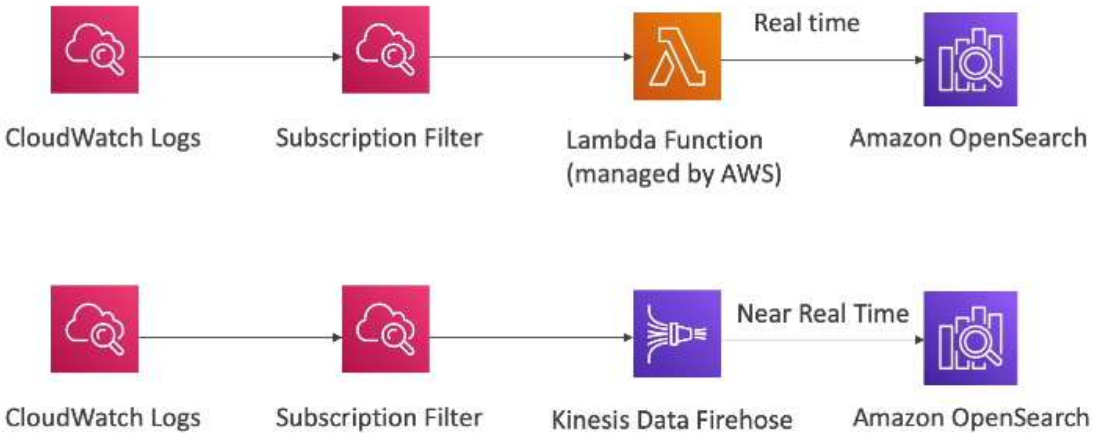
With **OpenSearch** we can search any field, even partially matches. It's common to use OpenSearch as a complement to another database. It does not natively support SQL (can be enabled via a plugin). It comes with OpenSearch Dashboards for visualization.

There are two modes: managed cluster and serverless cluster.

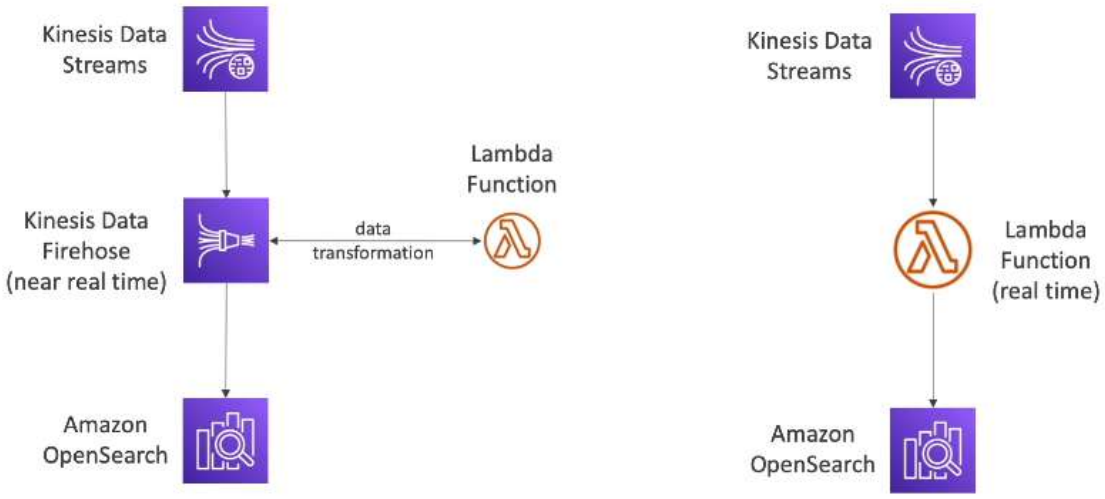
▼ **Solution Architecture - OpenSearch DynamoDB Integration.**



▼ **Solution Architecture - OpenSearch ClodWatch Logs Integration.**



▼ **Solution Architecture - OpenSearch KDS & KDF Integration.**



▼ ☑ **Amazon EMR.**

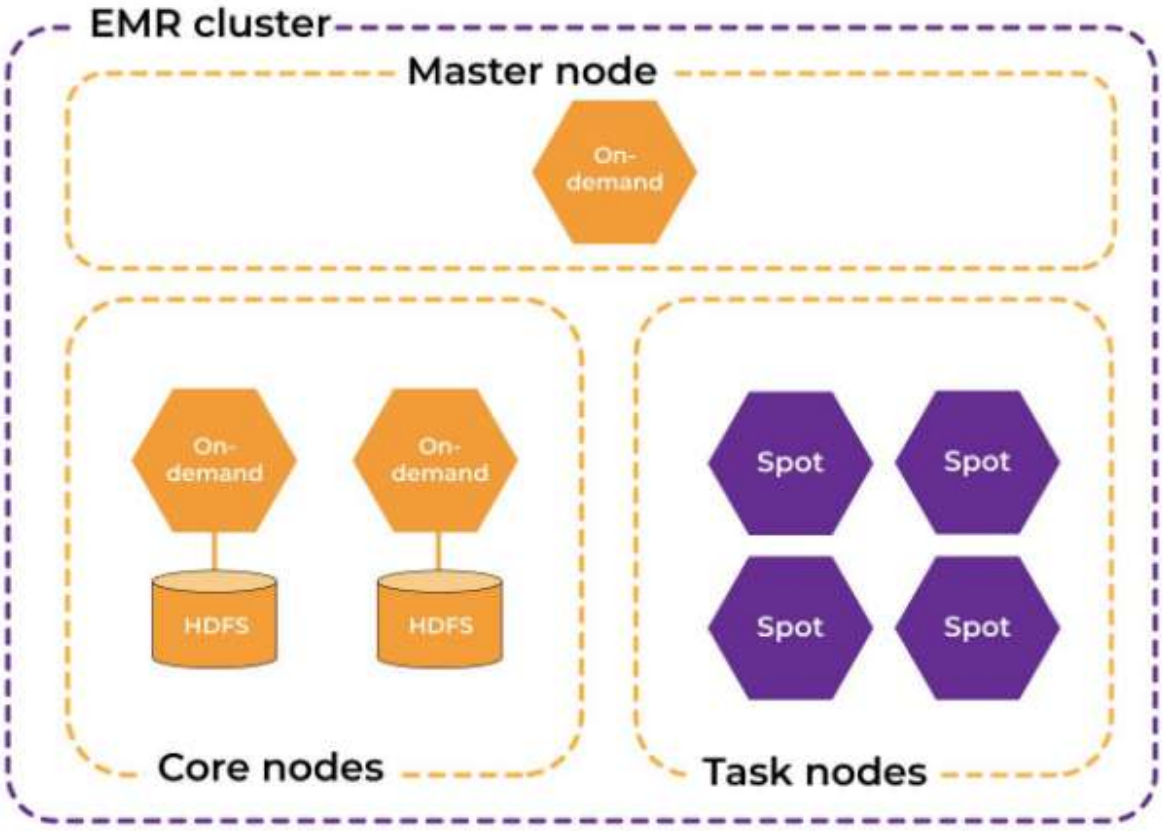
EMR helps creating Hadoop clusters (Big Data) to analyze and process vast amount of data. The clusters can be made of hundreds of EC2 instaces. It takes care of all the provisioning and configuration. It has auto-scaling and it's integrated with Spot instances.

EMR supports Apache Spark, HBase, Presto, Flink...

Usage: data processing, log analysis, ML, web indexing, big data ...

EMR Node Types

- **Master Node** - manages and coordinates the cluster (long running).
- **Core Node** - runs tasks and stores data (long running).
- **Task Node** (optional) - just to run tasks (short running).



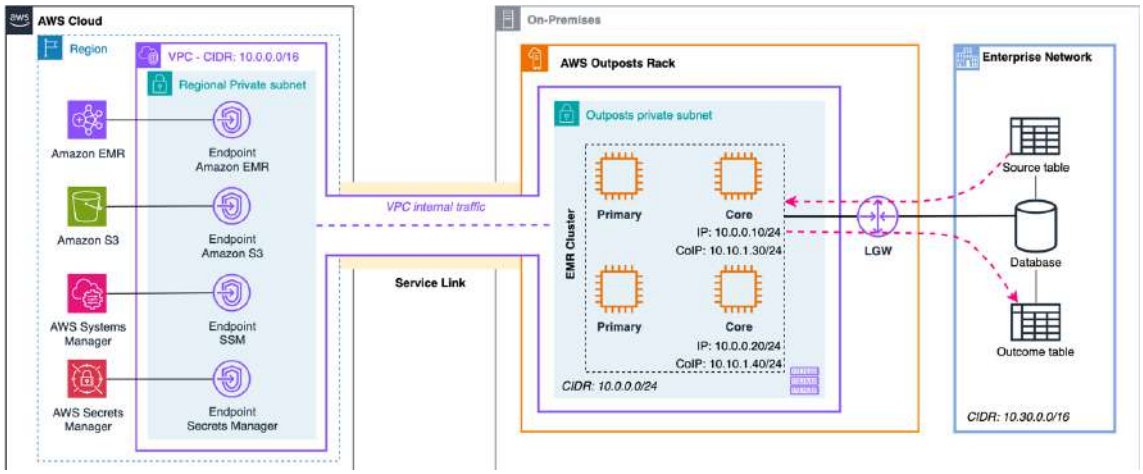
EMR Cluster Types

- **Long-running Cluster** - designed for ongoing data processing needs. They remain active for extended periods, allowing users to submit multiple jobs over time without the overhead of cluster startup and configuration.
- **Transient Cluster** - designed for short-lived, one-off jobs. They are created for specific processing tasks and terminated immediately after the job is complete.

Purchasing Options

- **On-demand** - reliable, predictable, wont be terminated.
- **Reserved** (min 1 year) - cost savings (EMR will automatically use if available).
- **Spot Instances** - cheaper, can be terminated, less reliable.

▼ **Solution Architecture - Process Data from an On-Premises Database.**

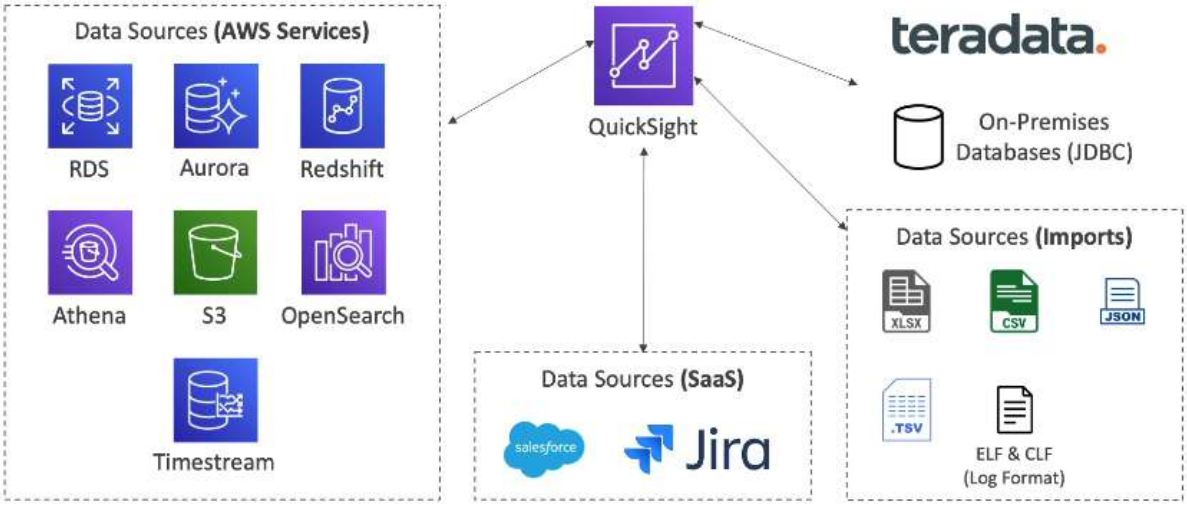


▼ ☑ **Amazon QuickSight.**

QuickSight is a serverless ML-powered business intelligence service to create interactive dashboards. It is fast, automatically scalable, embeddable, with per-session pricing. Can be integrated with RDS, Aurora, Athena, Redshift ...

It has in-memory computation capabilities using SPICE engine. In enterprise edition we have possibility to setup Column-Level Security (CLS) to prevent some columns to be displayed to users who dont have permission.

Usage: business analytics, building visualizations, perform ad-hoc analysis ...



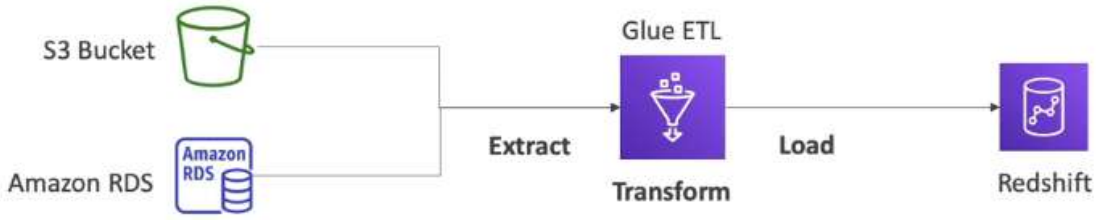
In QuickSight we can define users (standard versions) and groups (enterprise version). These users and groups only exist within QuickSight, not IAM!

Dashboard - read-only snapshot of an analysis that we can share. It preserves the configuration of the analysis.

We can share the analysis or the dashboard with users or groups. Users who see the dashboard can also see the underlying data.

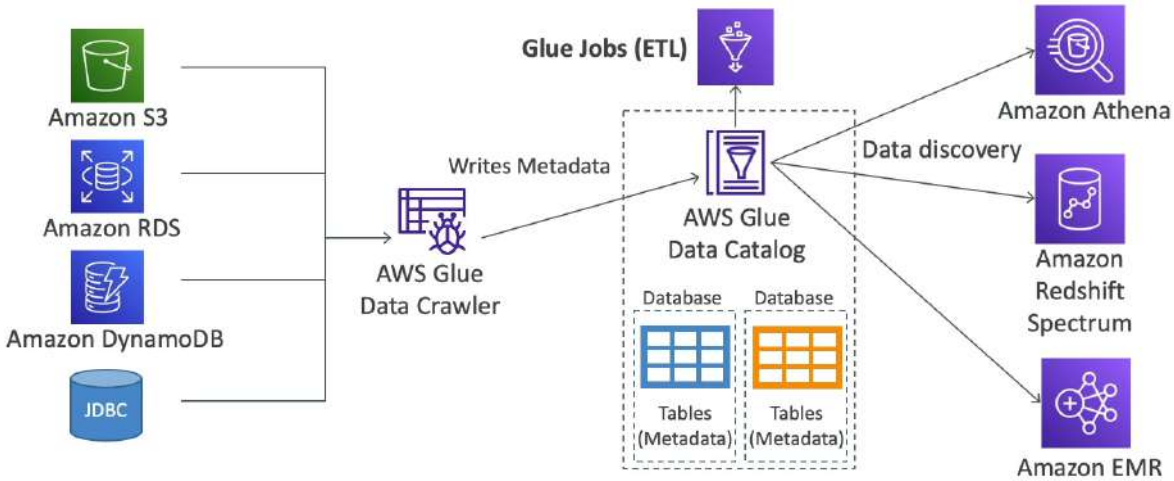
▼ ☐ **AWS Glue.**

AWS Glue is a serverless fully managed extract, transform, and load (**ETL**) service. It is useful for preparing and transforming data for analytics.



Glue Components

- **Data Catalog** - central repository that stores metadata about datasets in our AWS environment. Can be used by Athena, Redshift or EMR.
- **ETL Engine** - automatically generates Python or Scala code to extract data from various sources, transform it, and load it into target data stores.
- **Crawlers** - automated processes in AWS Glue that scan and infer the schema of data stored in various sources.
- **Glue Jobs** - ETL scripts that users create to transform data. These jobs can be scheduled and monitored within AWS Glue.



Glue Advanced Features

Glue Job Bookmarks - save and track the data that has already been processed, prevent re-processing old data.

Glue Elastic Views - combine and replicate data across multiple data stores using SQL. No custom code is needed, Glue monitors for changes in the source data (serverless).

Glue DataBrew - clean and normalize data using pre-built transformation.

Glue Studio - used to create, run and monitor ETL jobs in Glue.

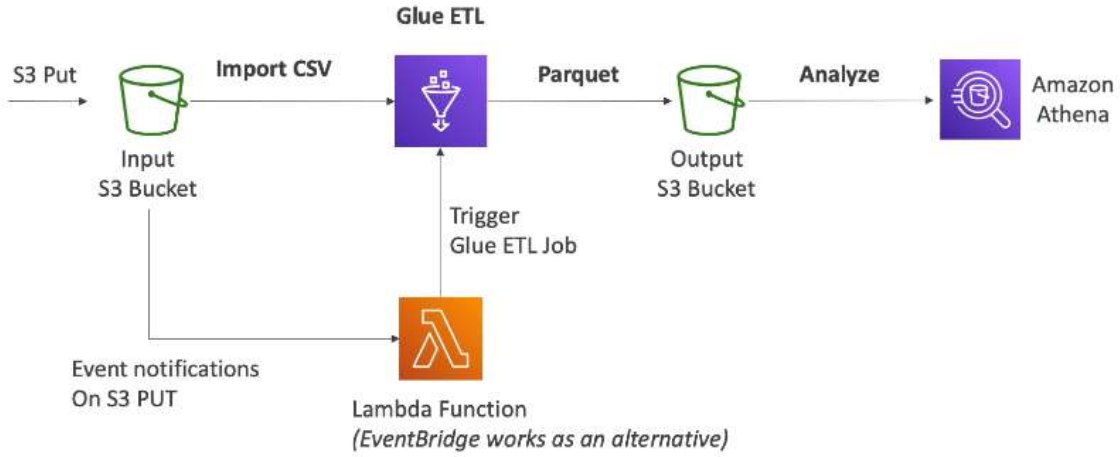
Glue Streaming ETL - run Glue jobs as streaming jobs. compatible with KDS, Kafka, MSK

Glue PySpark Jobs

PySpark provides distributed processing via Apache Spark, which makes Glue ideal for handling large datasets and complex transformations.

Feature	AWS Glue PySpark Jobs	AWS Lambda
Data Size	Large-scale data (GBs-TBs)	Small-medium data (up to a few GBs, constrained by memory and timeout limits)
Execution Time	Long-running jobs (up to hours)	Short-lived tasks (maximum 15 minutes)
Processing Type	Batch processing, complex ETL	Real-time, event-driven processing
Memory	Large memory and distributed computing available via Spark	Limited (up to 10 GB of memory)
Scalability	Automatically scales for large, distributed workloads	Automatically scales but limited by function constraints
Cost	Typically cost-effective for large datasets and long-running jobs	Cost-effective for short-lived, low-latency tasks
Integration with Other Services	Tight integration with S3, Redshift, RDS, Athena	Integration with many AWS services, especially for event-driven triggers

▼ **Solution Architecture - Convert data into Parquet format.**

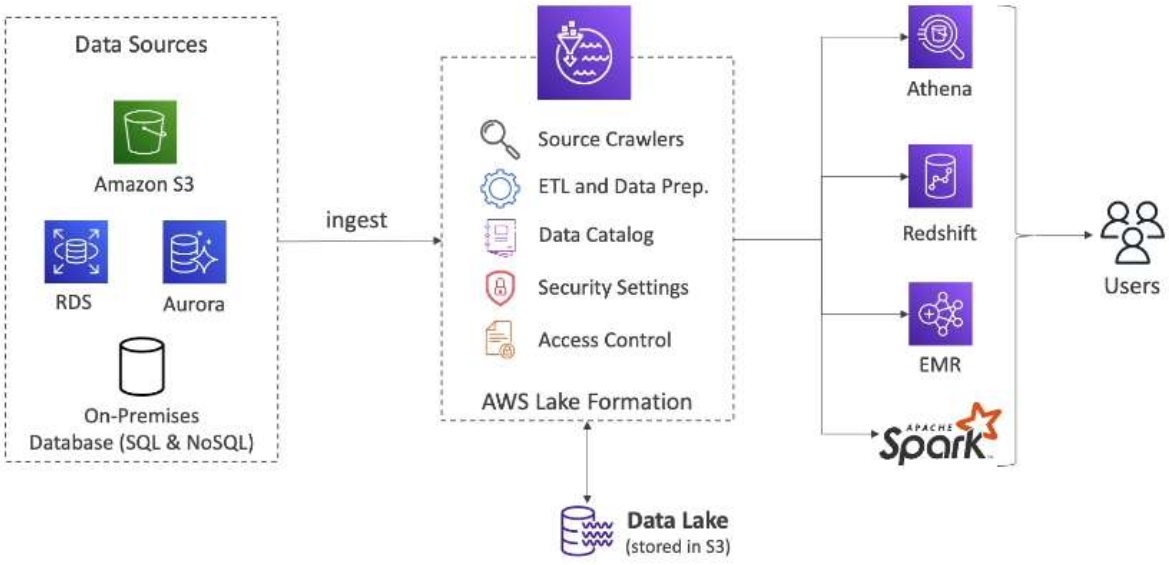


▼ ☑ **AWS Lake Formation.**

Lake Formation is a fully managed service that makes it easy to setup a data lakes in days. Data Lake is a central place to have all our data for analytics purposes. It is built on top of AWS Glue.

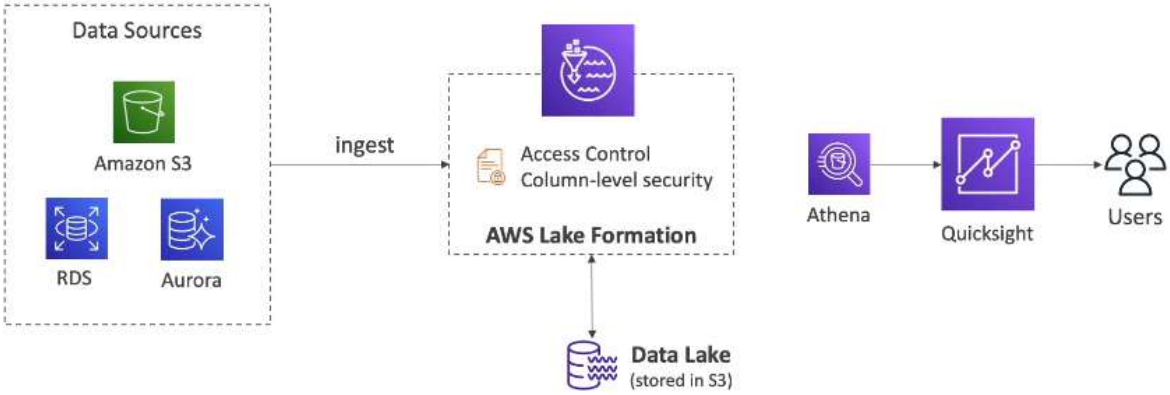
With Lake Formation we can discover, cleanse, transform and ingest data. It combines structured and unstructured data. It has blueprints for S3, RDS, relational & NoSQL DB, etc. Blueprints can help us migrate data from one place to the data lake.

We can have fine-grained access controls for our applications at the row and column level.



[Lake Formation - Centralized Permissions]

Lake Formation solves problem of security management by centralizing it. AWS Lake Formation simplifies the security of our data lake, making it easier to enforce data governance policies. We can consolidate data from multiple accounts into a single account.

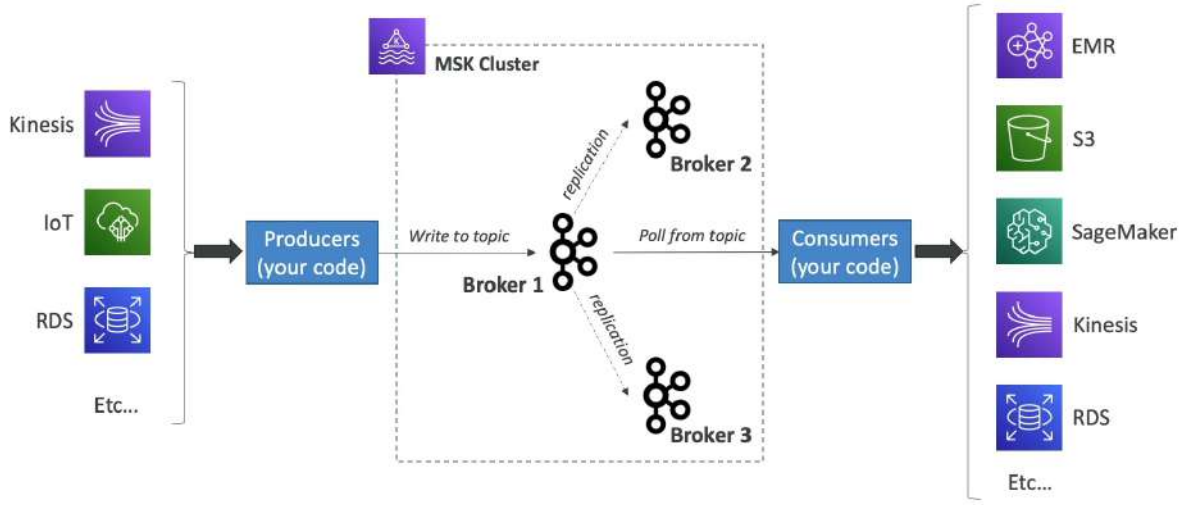


▼ ☑ MSK (Managed Streaming for Apache Kafka).

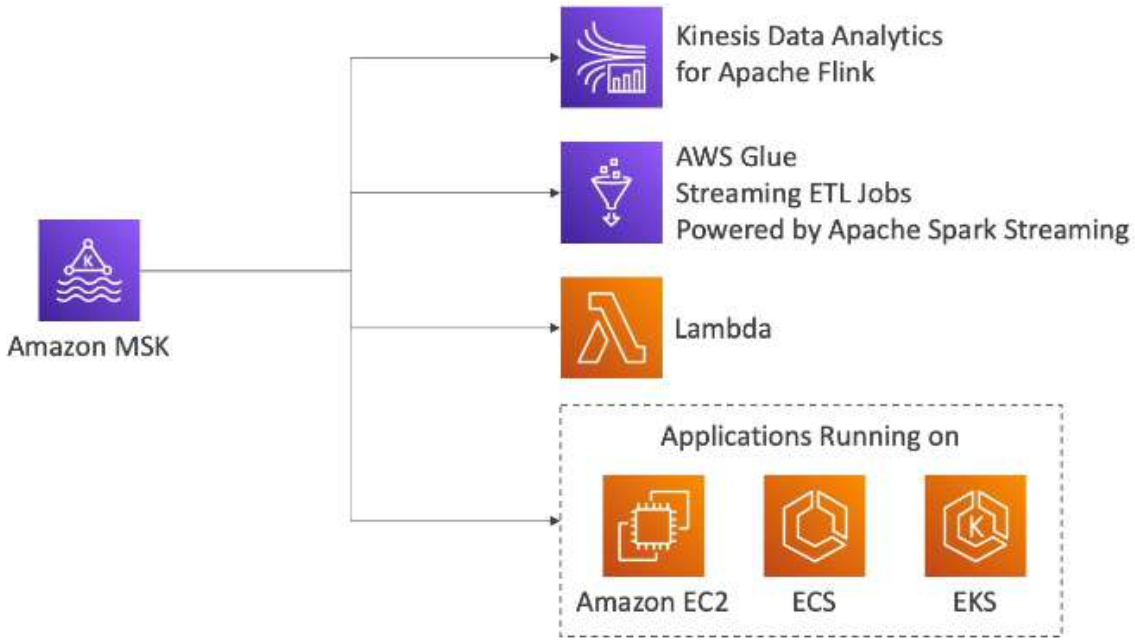
MSK is an alternative to Amazon Kinesis. It is fully managed Apache Kafka on AWS. It allows us to create, update and delete clusters.

MSK creates and manages Kafka brokers nodes and Zookeeper nodes for us. We can deploy the MSK cluster in our VPC and multi-AZ. Data is stored in **EBS** volumes for as long as we want.

MSK Serverless - run Apache Kafka on MSK without managing the capacity.



MSK Consumers

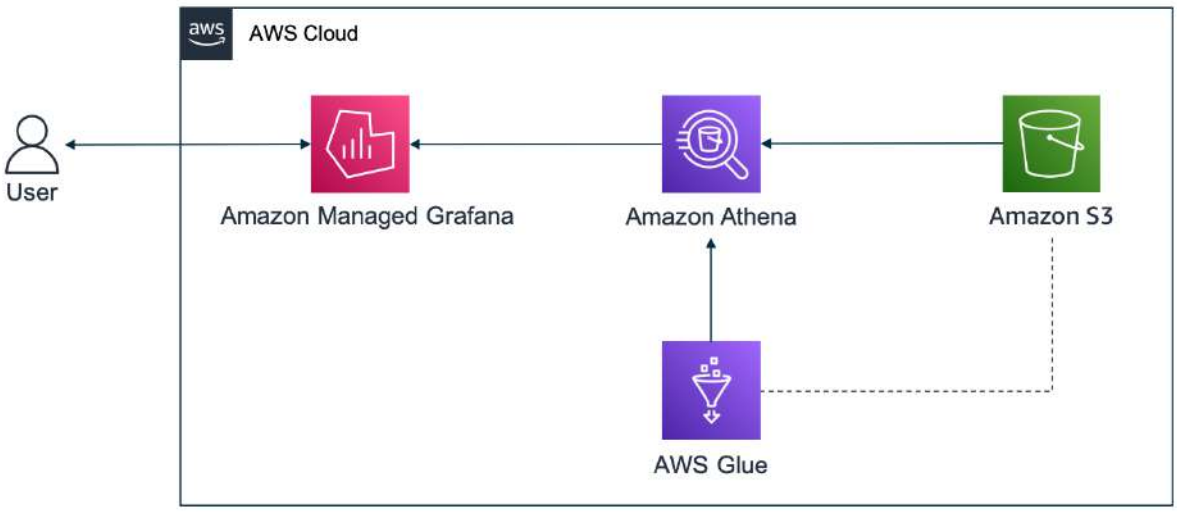


Kinesis Data Streams vs MSK

Feature	Amazon MSK (Managed Streaming for Apache Kafka)	Amazon Kinesis Streams
Technology	Apache Kafka	Kinesis-specific technology
Data Storage	Persistent storage with configurable retention	Configurable retention from 24 hours to 365 days
Sharding	Uses Kafka partitions for scaling	Uses shards for scaling
Message Ordering	Maintains order within a partition	Maintains order within a shard
Integration	Works with Kafka ecosystem tools and connectors	Integrated with AWS services like Lambda, S3
Operational Complexity	Requires understanding of Kafka operations	Simplified operation with AWS management
Cost	Based on broker instance types and storage	Based on data ingestion and retrieval

▼ ☑ **AWS Managed Grafana.**

AWS Managed Grafana is a fully managed service that makes it easy to deploy, operate, and scale Grafana for monitoring and observability. Grafana is a popular open-source analytics and monitoring tool that integrates with a wide variety of data sources to provide customizable dashboards and visualizations for metrics, logs, and traces. It scales automatically based on our needs.



AWS Managed Grafana integrates with IAM and supports SSO via AWS SSO and other identity providers. It is ideal for organizations that need to monitor a variety of data sources across different environments, including on-premises, cloud-native, and hybrid environments.

Machine Learning

▼ ☑ **ML AWS Services.**

- **Rekognition**

  Finds objects, people, text, secenes in images and videos using ML. Can be used for facial anaylsis and facial search to do user verification, people counting ...

- **Transcribe**

  Automatically converts speech to text using automatic speech recognition (ACR). It can automatically remove personally identifiable information using Redaction. Supports multi-lingual audio.

- **Polly**

  Turns text into speech using deep learning. It allows us to create applications that talk.

- **Translate**

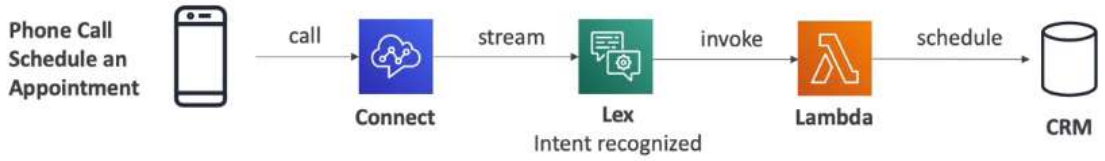
It provides us natural and accurate language translation. Allows us to **localize content** - such as websites and applications for international users.

- **Lex**

  Same technology that powers Alexa. It's used for ASR and NLP, helps in building chatbots and call center bots.

- **Connect**

  Used to receive calls, create contact flows, cloud-based virtual contact center. No upfront payments, 80% cheapter than traditional contact center solutions.

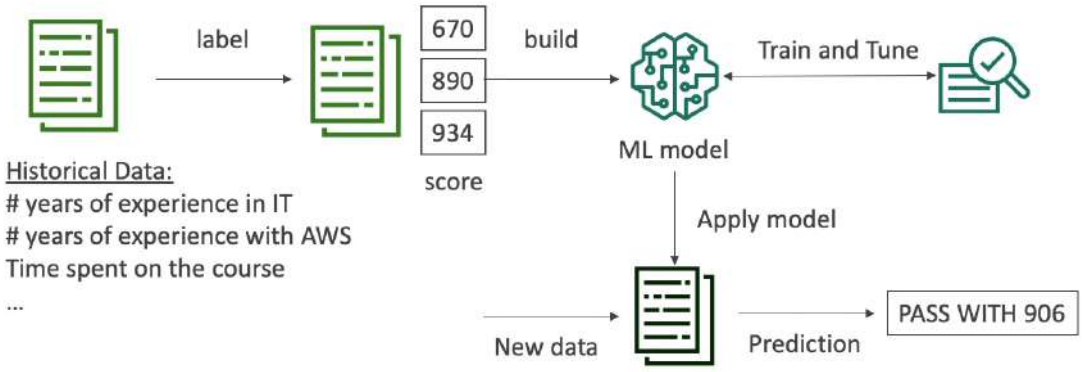


- **Comprehend**

  Used for NLP, fully managed and serverless service. Used for extracting key phrases, places, people ... Analyzes text using tokenization and parts of speech, andautomatically organizes a collection of text files by topic.

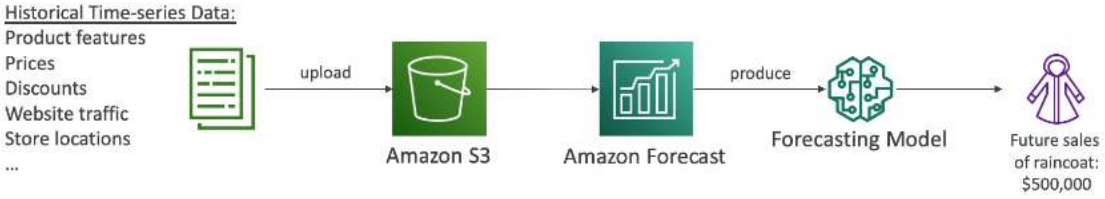
- **SageMaker**

  Fully managed service for developers and data scientists to build ML models.



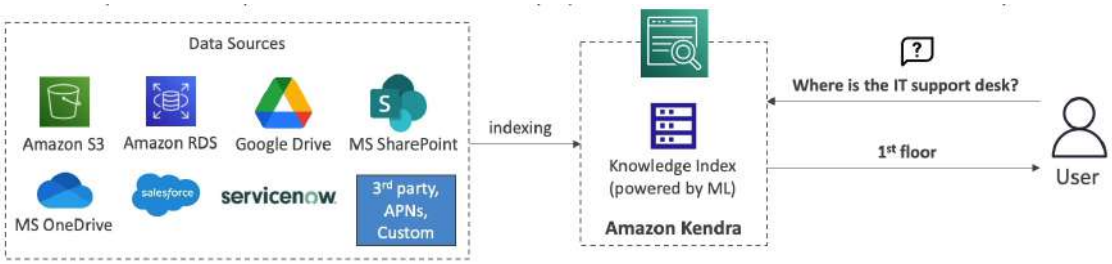
- **Forecast**

Fully managed service that uses ML to deliver highly accurate forecasts. It's 50% more accurate than looking at the data itself.



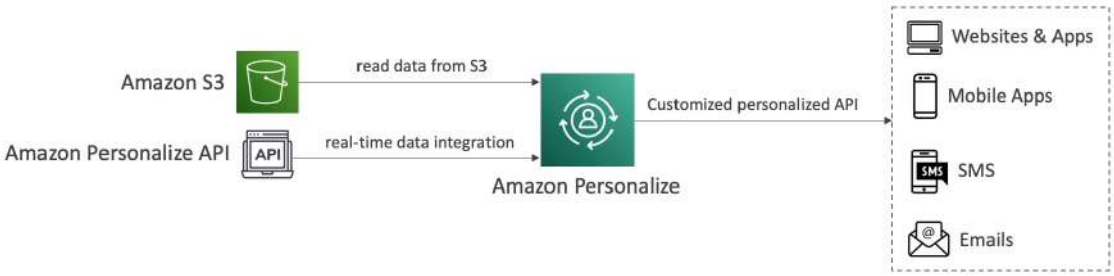
- **Kendra**

  Fully managed document search service powered by ML. It extracts answers from within a documents (text, pdf, HTML ...). It can learn from user interactions/feedback to promote preferred results (Incremental Learning).



- **Presonalize**

  Fully managed ML service to build apps with real-time personalized recommendations. Can be integrated into existing websites and applications.



- **Textract**

  Automatically extracts text, handwriting and data from any scanned documents using ML.

- **Monitron**

Detects abnormal conditions in industrial machinery, enabling us to implement predictive maintenance and reduce unplanned downtime.

- **Panorama**

  Aims to bring computer vision (CV) capabilities to the edge devices in industrial environments.

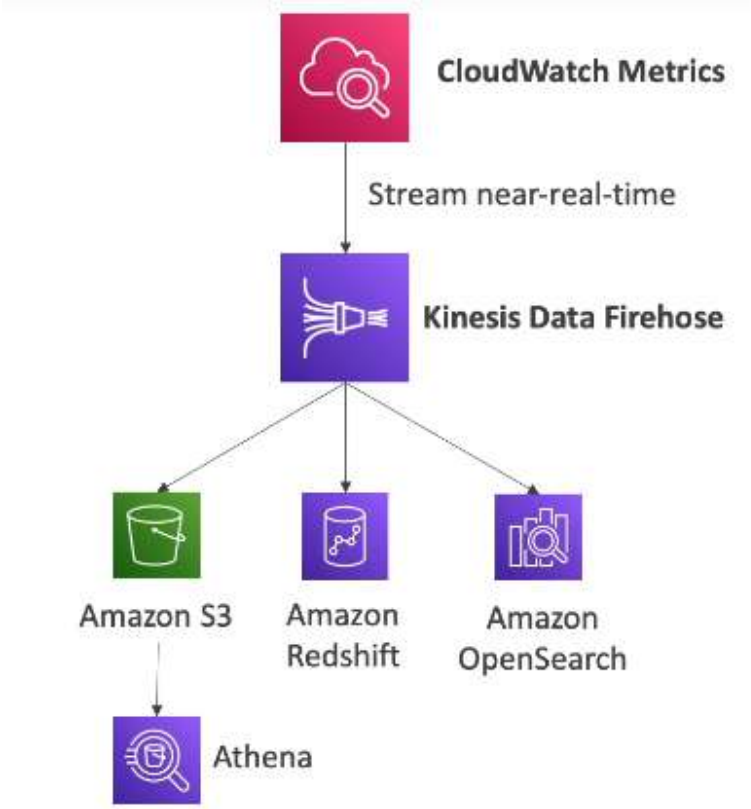
AWS Monitoring & Audit

▼ ☑ **CloudWatch Metrics.**

CloudWatch provides metrics for every service in AWS. Metric is a variable to monitor (CPU Utilization, Network In, etc). We can create custom metrics too (**Memory Utilization**).

Each metric has timestamps and dimension. **Dimension** is an attribute of a metric (instance id, environment, etc.). CloudWatch can create dashboards of metrics.

CloudWatch Metric Streams - continually stream CloudWatch metrics to a destination of our choice, with near real-time delivery and low latency. We can use KDF or third party service provider (Datadog, Dynatrace, etc). We have option to filter metrics to only stream a subset of them.



▼ ☐ **CloudWatch Logs.**

CloudWatch Logs is a logging service that monitors, stores, and access log files from AWS resources, applications, and services. Logs are encrypted by default.

- Log groups - arbitrary name, usually representing an application.
- Log stream - instance within applications/log files/containers.

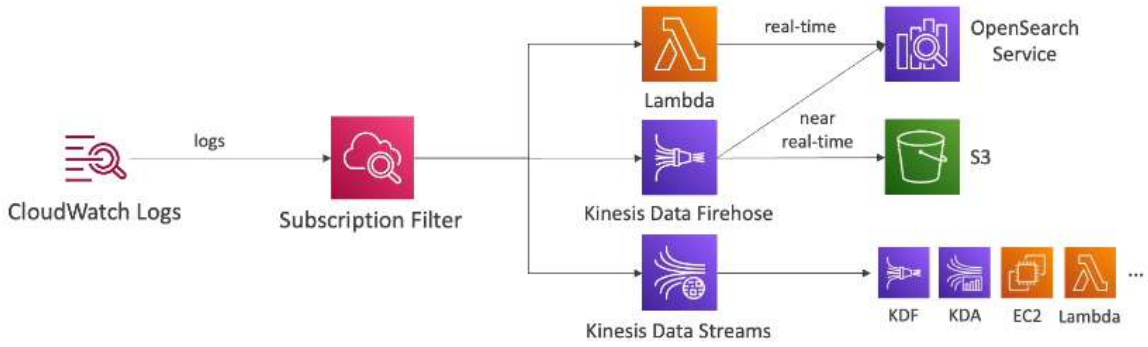
CloudWatch Logs can send logs to: S3, KDS, KDF, AWS Lambda, OpenSearch.

Logs can be collected from: Elastic Beanstalk, ECS, AWS Lambda, CloudTrail, CloudWatch Log agents (on EC2 instances or on-premises servers), Route53.

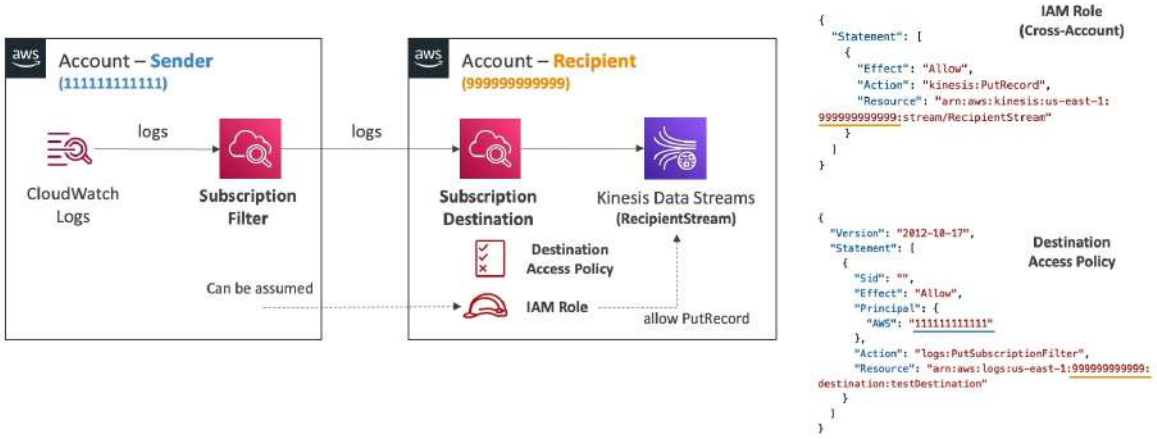
CloudWatch Logs Insights - search and analyze log data stored in CloudWatch Logs. Provides a purpose-built query language. It automatically discovers fields from AWS services and JSON log events. We can save queries and add them to CloudWatch Dashboard. It's query engine, not a real-time engine.

Log data can take up to 12h to become available for export. It is not near real-time or real-time, if we want that we should use Logs Subscriptions instead.

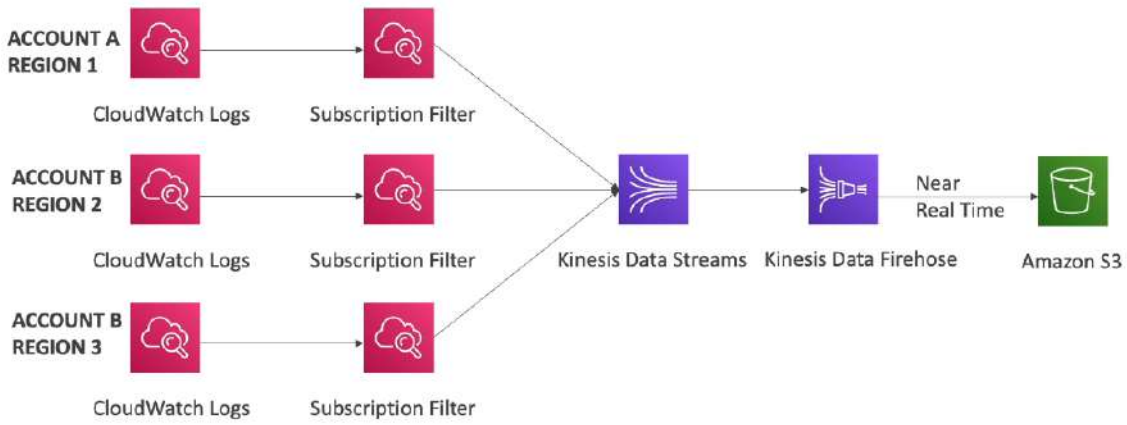
CloudWatch Logs Subscriptions - lets us get a real-time log events from CloudWatch Logs for processing and analysis. We can send to KDS, KDF or AWS Lambda. There is Subscription Filter which can filter which logs will be delivered to our destination.



We can have Cross-Account Subscription which can send log events to resources in different AWS account (KDS, KDF).

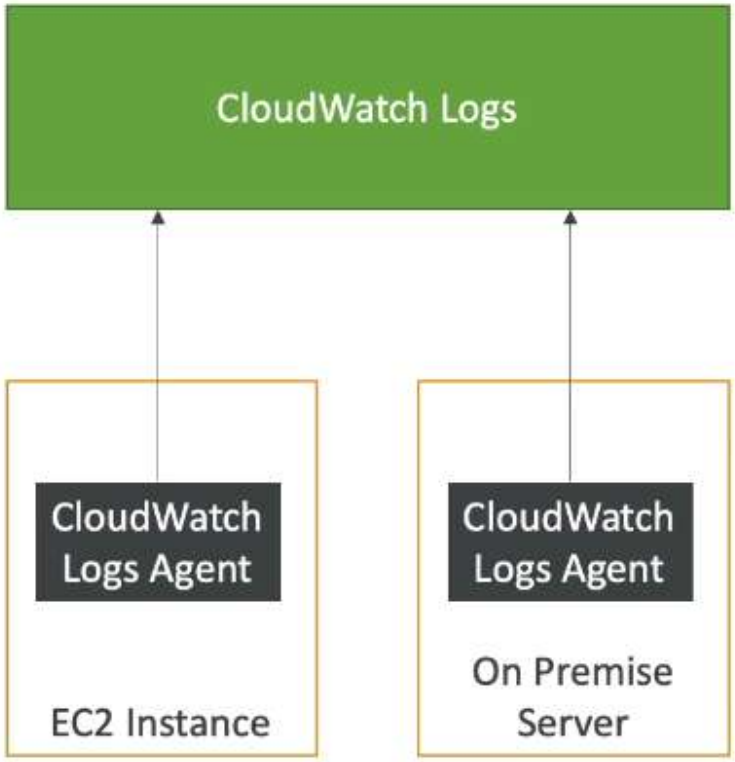


▼ **Solution Architecture - Logs Aggregation (Multi-Account & Multi Region).**



▼ ☑ **CloudWatch Agent & CloudWatch Logs Agent.**

By default, no logs from our EC2 instance will go to CloudWatch, we need to run a CloudWatch Agent on EC2 to push the log files we want (need to make sure IAM permissions are correct).



CloudWatch Logs Agent - old version of the agent. Can only send to CloudWatch Logs.

CloudWatch Unified Agent - it collects additional system-level metrics such as RAM, processes, etc. It collect logs to send them to CloudWatch Logs. It has

centralized configuration using SSM Parameter Store.

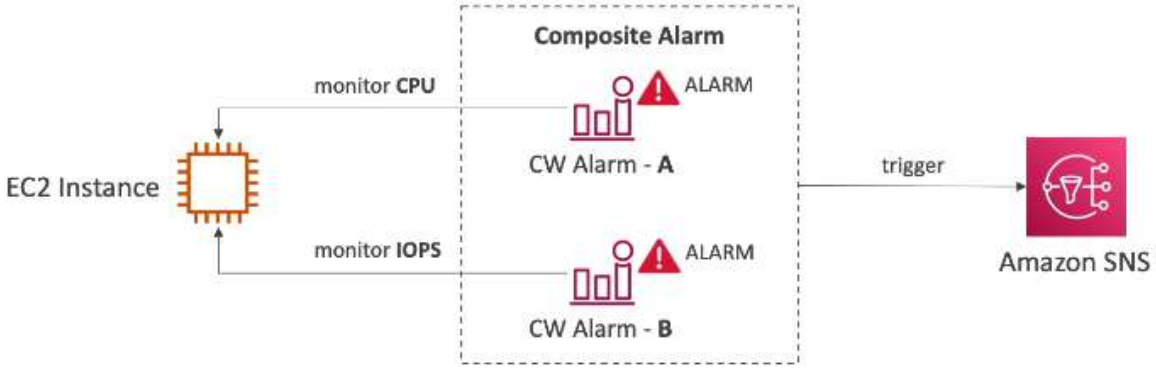
▼ ☐ **CloudWatch Alarms.**

CloudWatch Alarms - used to trigger notifications for any metrics. It has various options (sampling, percentage, max, min, etc). With CloudWatch Alarm we can stop, terminate, reboot or recover an EC2 instance, trigger auto scaling action or send notification to SNS.

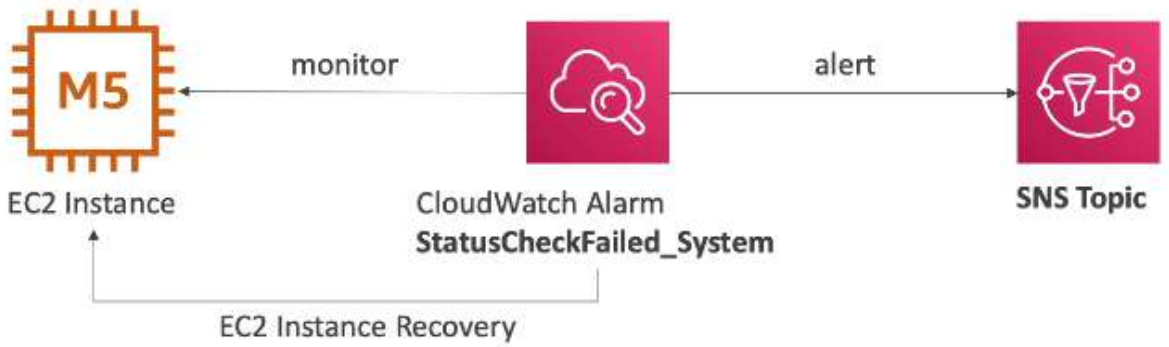
Alarm States: OK, INSUFFICIENT_DATA, ALARM.

Period - length of time in seconds to evaluate the metrics.

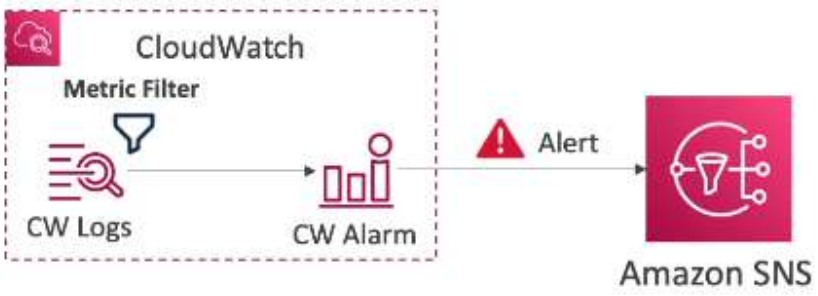
Composite Alarms - monitoring the states of multiple other alarms and we can use AND and OR conditions to merge them.



There is a status check to check the EC2 and we can define a CloudWatch Alarm to monitor a specific EC2 instance. In case the alarm is being breached, we can start an EC2 instance recovery or send an alert to our SNS topic.



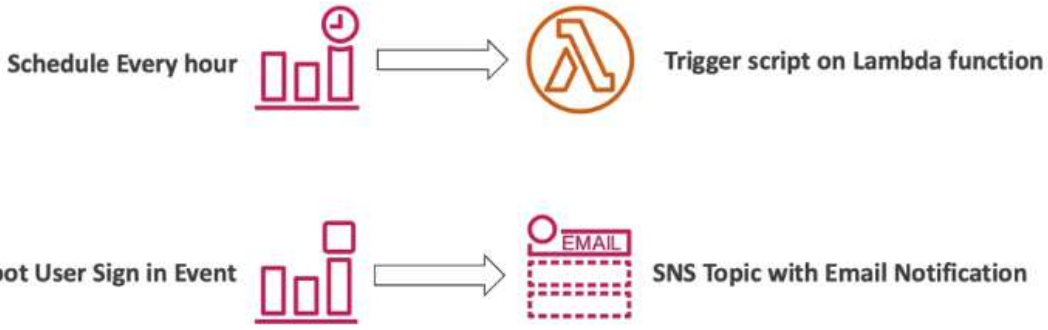
Alarms can be created based on CloudWatch Logs Metrics Filters.

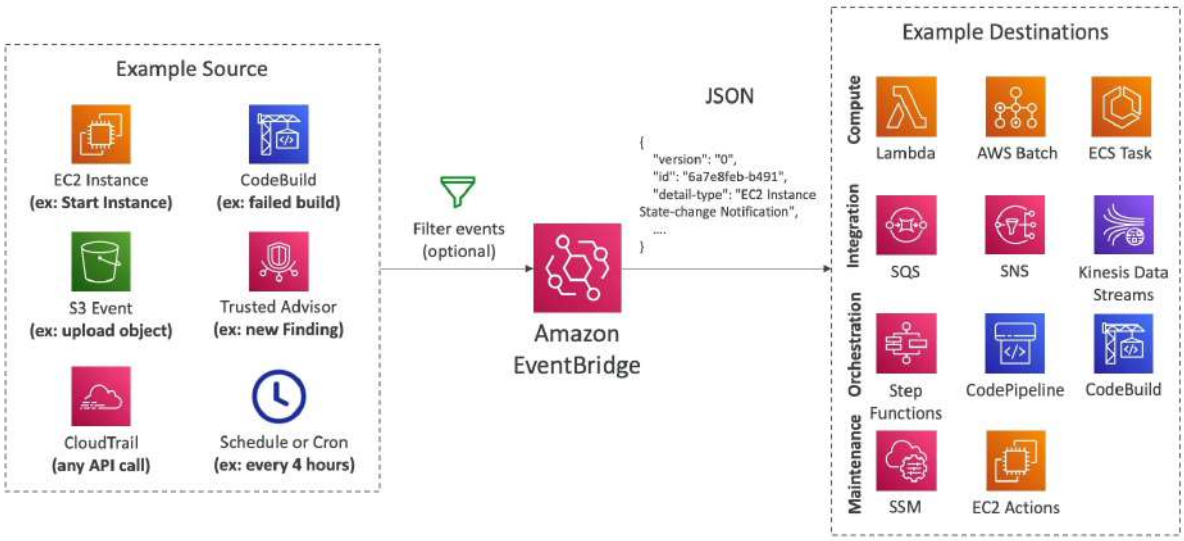


▼ ☐ **CloudWatch Events (EventBridge).**

EventBridge is a serverless event bus service that makes it easy to connect applications using data from our own applications, integrated SaaS applications, and AWS services.

We can schedule events with CRON jobs or we can make event rules to react to a service doing something. Also it's possible to trigger Lambda functions, send SQS/SNS messages ...





EventBridge Schema Registry - allows us to generate code for our application, that will know in advance how data is structured in the event bus. Schema can be versioned.

Event Bus Types

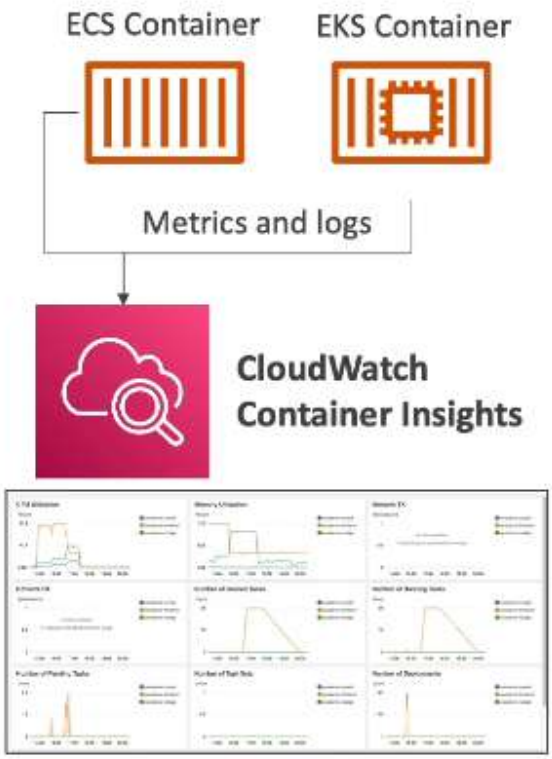
Event buses can be accessed by other AWS accounts using resource-based policies.

Resource-based Policy - manages permissions for a specific Event Bus. It is used to aggregate all events from our AWS Organization in a single AWS account or AWS region.

We can archive events sent to an event bus (indefinetely or set period) and we can replay archived events.

▼ ☐ **CloudWatch Insights & Operational Visibility.**

CloudWatch Container Insights - used to collect, aggregate and summarize metrics and logs from containers. In EKS and Kubernetes, CloudWatch Insights is using a containered version of the CloudWatch Agent to discover containers. We cant use it on-premises.



CloudWatch Lambda Insights - monitoring and troubleshooting solution for serverless applications running on AWS Lambda. It collects, aggregates and summarizes system-level metrics and diagnostic information (cold starts, Lambda worker shutdowns).

CloudWatch Contributor Insights - analyzes log data and creates time series that display contributor data. We can see metrics about the top-N contributors. It helps us to find top talkers and understand who or what is impacting system performance. We can build our rules from scratch or we can also use sample rules that AWS has created.

Use cases: find bad hosts, identify the heaviest network users or find the URLs that generate the most errors.

CloudWatch Application Insights - provides automated dashboards that show potential problems with monitored applications to help isolate ongoing issues. It is powered by SageMaker.

It gives us enhanced visibility into our applicaion health to reduce the time it will take us to troubleshoot and repair our applications. Findings and alerts are sent to EventBridge and SSM OpsCenter.

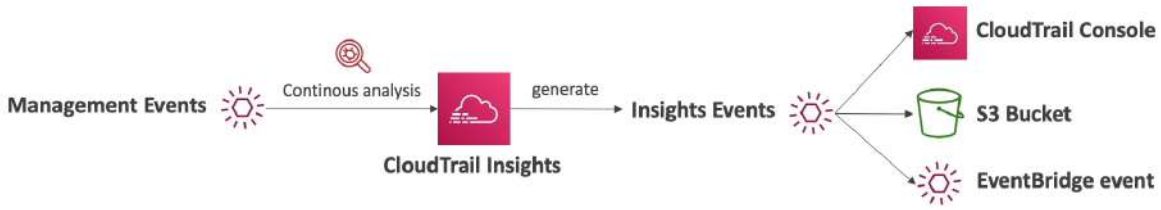
▼ ☑ CloudTrail.

AWS CloudTrail provides governance, compliance and audit for our AWS account. It is enabled by default. We can get history of events and API calls made within our account. It's possible to put logs from CloudTrail into CloudWatch Logs or S3. Can be applied to all regions (default) or a single region. By default, CloudTrail event log files are encrypted using Amazon S3 server-side encryption (SSE).

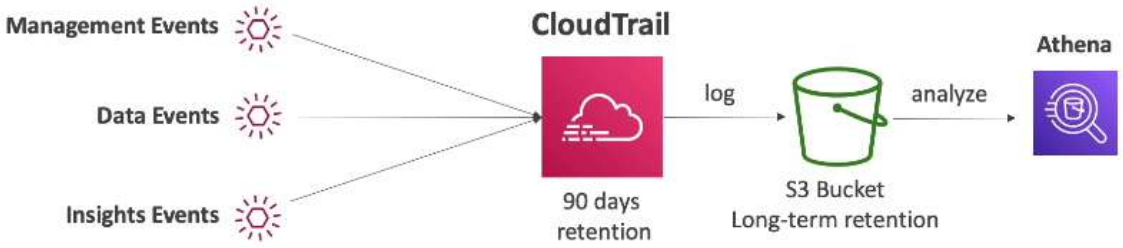
CloudTrail Events

- **Management Events** - operations that are performed on resources in our AWS account. By default, trails are configured to log management events. We can separate read events from write events.
- **Data Events** - S3 object-level activity and Lambda function execution activity (get, delete, put). By default, data events are not logged because high volume operations. It can separate read events from write events.
- **CloudTrail Insights Events**

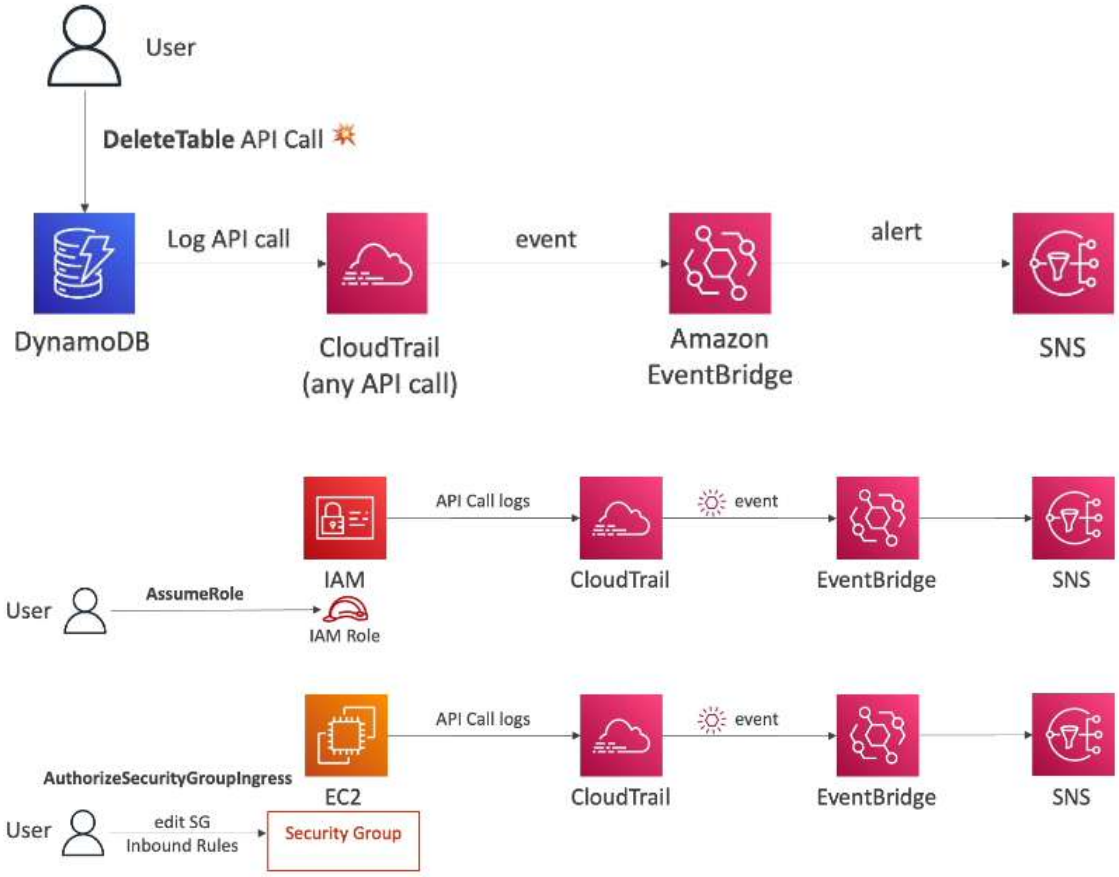
CloudTrail Insights is a feature within AWS CloudTrail that helps us detect unusual or unexpected activity in your AWS accounts. It continuously analyzes our CloudTrail event logs to identify patterns and anomalies, enabling us to quickly respond to potential security issues.



Events are stored for 90 days in CloudTrail. To keep events beyond this period we should log them to S3 and use Athena.



▼ **Solution Architecture - Intercept API Calls (CloudTrail + EventBridge)**



▼ **☑ AWS Config.**

AWS Config helps us with auditing and recording compliance of our AWS resources. It can record configurations and changes over time. It is possible to store the configuration data into S3 (analyzed by Athena).

Questions that can be solved by AWS Config:

- Is there unrestricted SSH access to my security groups?

- Do my buckets have any public access?
- How has my ALB configuration changed over time?

Config is a per-region service and can be aggregated across regions and accounts. We can receive alerts (SNS notifications) for any changes.

Config Rules

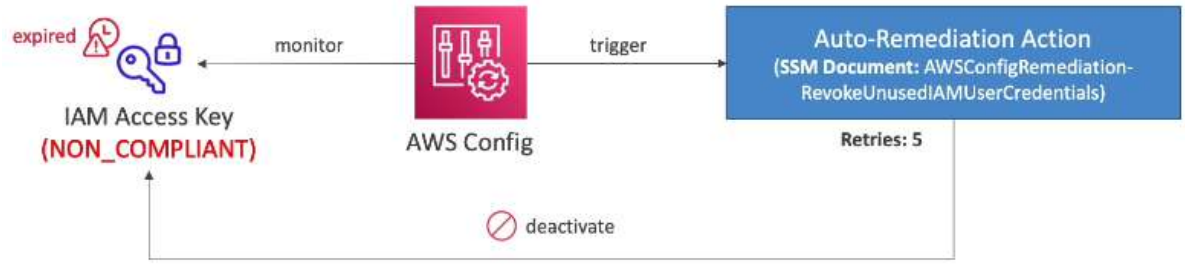
We can use AWS managed config rules or make custom config rules (**must be defined in AWS Lambda**). Rules can be evaulated/triggered for each config change and/or at regular time intervals.

Config Rules does not prevent actions from happening (**no deny**).

Config Remediations

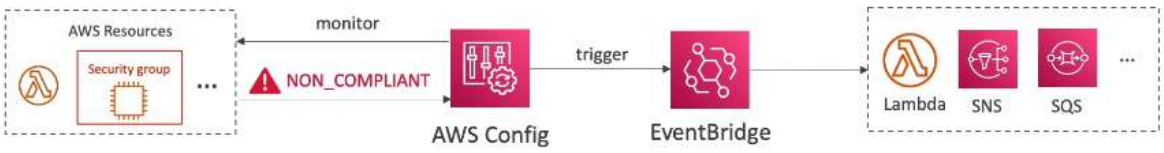
We can automate remediation of non-compliant resource using SSM Automation Documents. It is possible to create custom Automation Documents that invokes Lambda function.

We can set Remediation Retries if the resource is still non-compliant after auto-remediation.

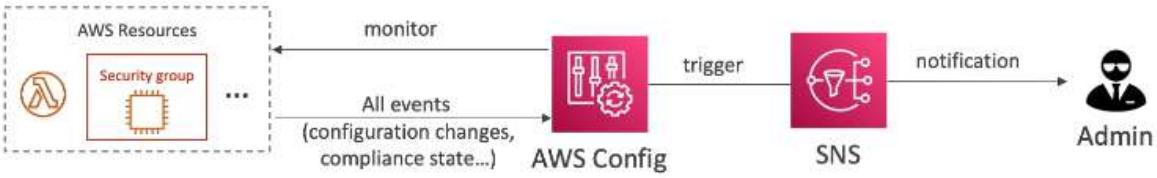


Config Notifications

We can use EventBridge to tigger notifications when AWS resources are non-compliant.



There is ability to send configuration changes and compliance state notifications to SNS.



▼ ☑ **AWS Health Dashboard.**

AWS Health Dashboard is a tool that provides personalized views into the status and health of our AWS services and resources. It helps us monitor and manage the health of our AWS infrastructure by delivering alerts and notifications about service disruptions, scheduled maintenance, and other impactful events.

- **Service Health** is the single place to learn about the availability and operations of AWS services. It offers the possibility to subscribe to an RSS feed to be notified of interruptions to each service.

  Service History - displays the status of AWS services across all regions. It shows the current operational status and any ongoing issues.

- **Your Account (Personal Health Dashboard (PHD))** - offers a personalized view into the health of AWS services that are specifically relevant to our AWS environment. It provides alerts and remediation guidance when AWS experiences events that may impact our resources. Shows how AWS outages directly impact us and our AWS resources.

IAM - Advanced

▼ ☑ **AWS Organizations.**

AWS Organizations is a global service that allows us to manage multiple AWS accounts. The main account is called master account. There is API which automates AWS account creation.

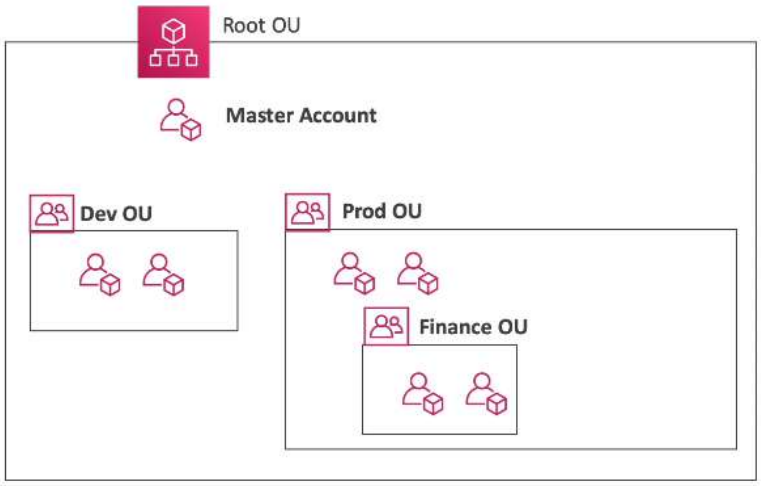
Cost benefits:

- Consolidated Billing across all accounts - we receive a single invoice for all the accounts in our organization, simplifying the billing process.
- Pricing benefits from aggregated usage - higher usage typically results in lower per-unit costs, providing significant savings.
- Pooling of reserved EC2 instances for optimal savings.

Multi Account Strategies

We can create accounts per department, per cost center, per dev/test/prod, for better resource isolation, to have separate per-account service limits, isolated account for logging ...

Good practice is to enable CloudTrail on all accounts and send logs to central S3 account, also send CloudWatch logs to central logging account.

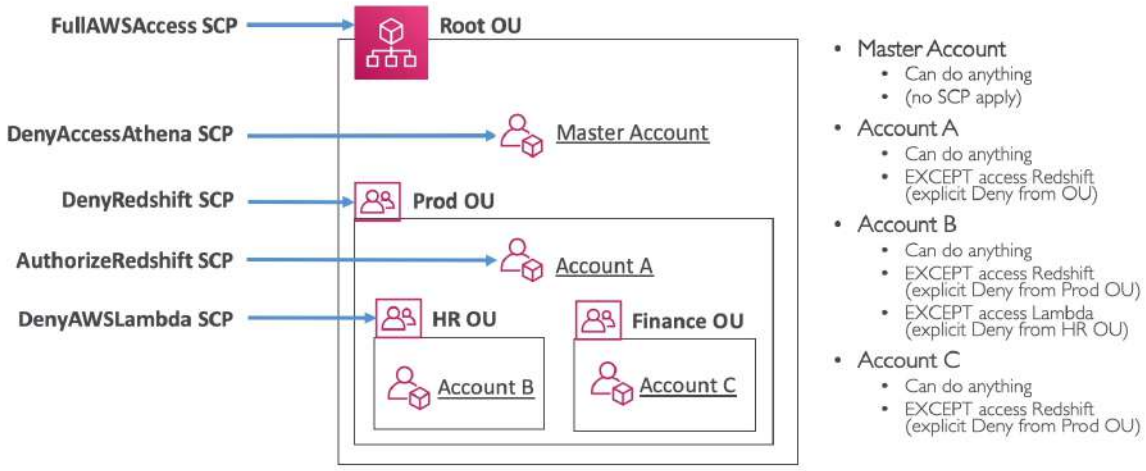


Service Control Policies (SCPs) - allows us to control the services and actions that users and roles can access within the accounts of our organization (whitelist or blacklist IAM actions).

Can be applied to the OU (organizational unit) or account level (does not apply to the master account). It is applied to all the users and roles of the account, including root. SCPs are inherited down the hierarchy. An SCP attached to a root or OU will affect all accounts and OUs beneath it.

It will not affect service-linked roles (service-linked roles enable other AWS services to integrate with organizations and cant be restricted).

SCP must have an explicit allow (does not allow anything by defult).



▼ ☑ **IAM Advanced Policies.**

aws:SourceIp - restrict the client IP from which the API calls are being made.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*",
        "Condition": {
            "NotIpAddress": {
                "aws:SourceIp": [
                    "192.0.2.0/24",
                    "203.0.113.0/24"
                ]
            }
        }
    }
}
```

aws:RequestedRegion - restrict the region the API calls are made to.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "DenyAllOutsideRequestedRegions",
            "Effect": "Deny",
            "NotAction": [
                "cloudfront:*",
                "iam:*",
                "route53:*",
                "support:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestedRegion": [
                        "eu-central-1",
                        "eu-west-1",
                        "eu-west-2",
                        "eu-west-3"
                    ]
                }
            }
        }
    ]
}
```

ec2:ResourceTag - restrict based on tags.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:startInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:us-east-1:12345:instance/
            "Condition": {
```

```
                "StringEquals": {
                    "ec2:ResourceTag/Project": "DataAnalytics"
                    "aws:PrincipalTag/Department": "Data"
                }
            }
        }
    ]
}
```

aws:MultiFactorAuthPresent - force MFA.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "ec2:StopInstances",
      "ec2:TerminateInstances"
    ],
    "Resource": ["*"],
    "Condition": {
        "Bool": {
            "aws:MultiFactorAuthPresent": "true"
          }
         }
  }]
}
```

IAM for S3

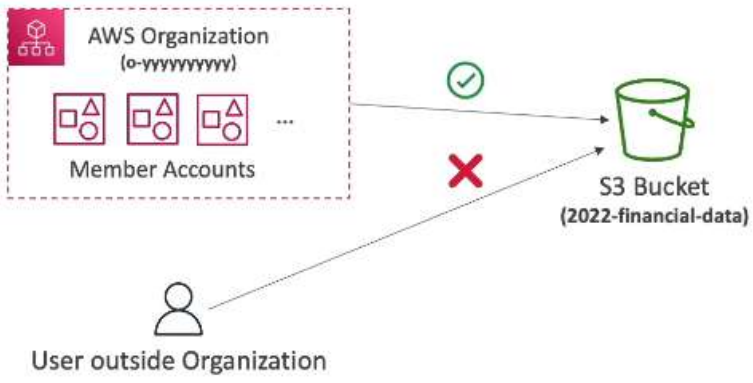
s3:ListBucket - applies to arn:aws:s3:::test (**bucket level permission**).

s3:GetObject, **s3:PutObject**, **s3:DeleteObject** - applies to arn:aws:s3:::test/* (**object level permission**).

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "AllowAuroraToExampleBucket",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:AbortMultipartUpload",
                "s3:ListBucket",
                "s3:DeleteObject",
                "s3:GetObjectVersion",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*",
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

aws:PrincipalOrgID - can be used in any resource policies to restrict access to accounts that are member of an AWS Organization.




```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject"
        ],
        "Resource": "arn:aws:s3:::2022-financial-data/*",
        "Condition": {
          "StringEquals": {
            "aws:PrincipalOrgID": "o-sabhong3hu"
          }
        }
      }
    ]
}
```

▼ ☑ **IAM Resource-based Policies vs IAM Roles.**

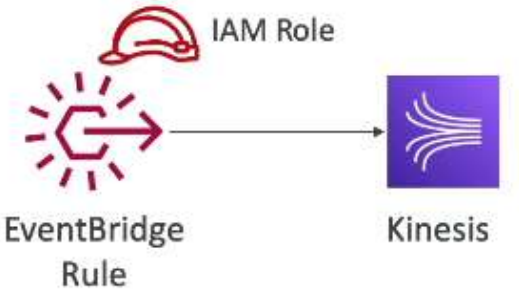
When we want to perform an API call on a S3 bucket cross account we have two options. Attaching a resource-based policy to a resource (e.g. S3 bucket policy) or using a role as a proxy.

- When we assume a role (user, application or service), we give up our original permissions and take the permissions assigned to the role.
- When using a resource-based policy, the principal doesnt have to give up his permissions.

> User in account A needs to scan a DynamoDB table in account A and dump it in an S3 bucket in account B. ⇒ we should use resource-based policy.

It is really important when we use EventBridge. When rule runs, it needs permissions on the target.

We have targets that support resource-based policy (**AWS Lambda, SNS, SQS, S3 buckets, API Gateway**) and targets that supports IAM role (**Kinesis stream, SSM Run Command, ECS task**).



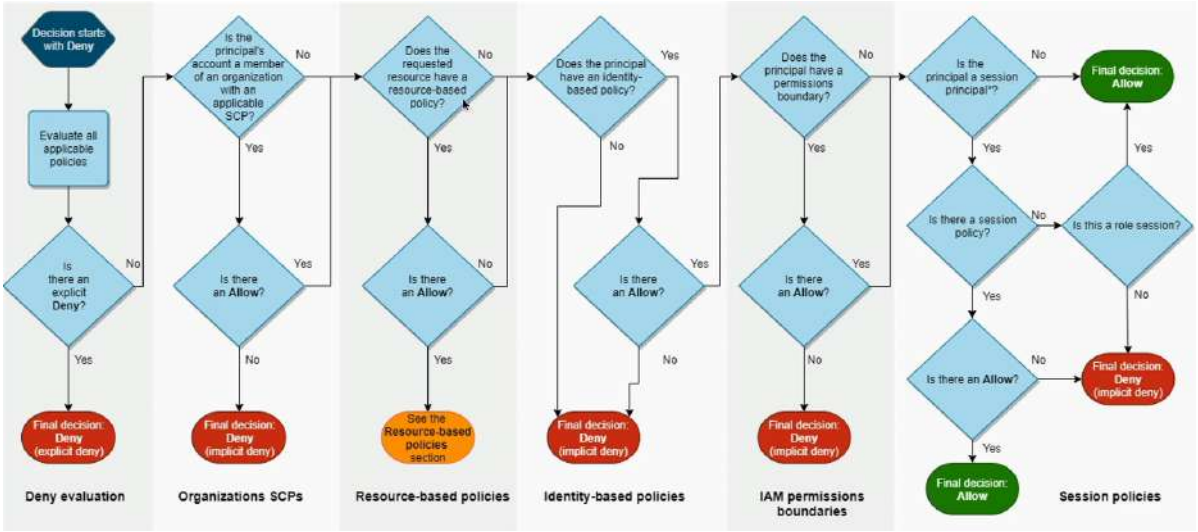
▼ ☑ **IAM Policy Evaluation Logic.**

IAM Permission Boundaries - advanced feature to use a managed policy to set the maximum permissions an IAM entity can get. They are supported for users and roles (not groups). They can be used in combinations of AWS Organizations SCP.

- Can delegate responsibilites to non administrators within their permissions boundaries.
- Allow developers to self-assign policies and manage their own permissions, while making sure they cant escalate their privileges.

- Useful to restrict one specific user (instead of a whole account using Organization & SCP).

IAM Policy Evaluation Logic



- Can you perform sqs:CreateQueue? NO
- Can you perform sqs:DeleteQueue? NO
- Can you perform ec2:DescribeInstances? NO

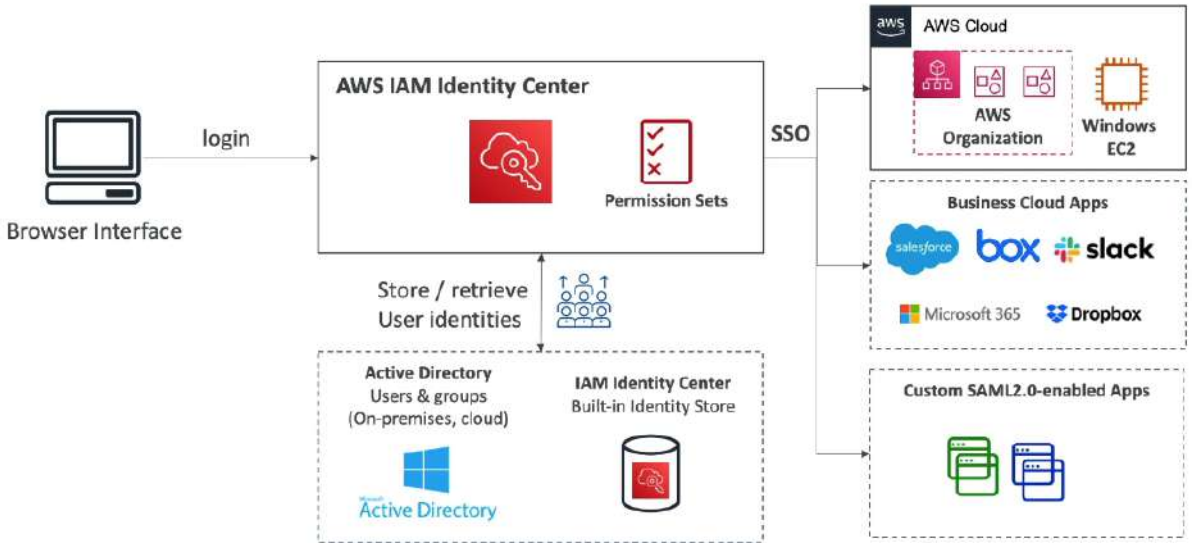
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "sqs:*",
      "Effect": "Deny"
      "Resource": "*"
    },
    {
      "Action": [
        "sqs:DeleteQueue"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

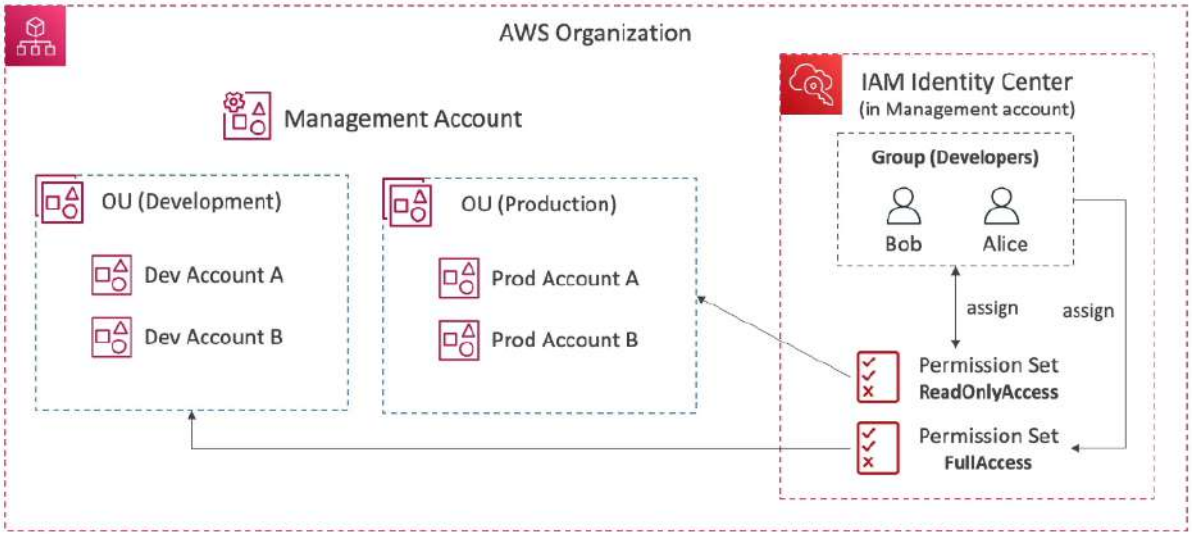
Explicit Deny

▼ ☐ AWS IAM Identity Center.

AWS IAM Identity Center (AWS Single Sign-On (AWS SSO)) is a cloud-based service that simplifies managing access to multiple AWS accounts in AWS Organizations and business applications.

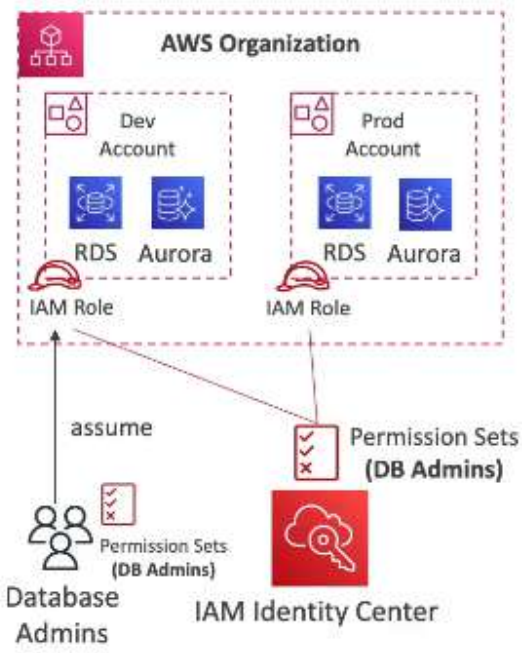
An **Identity Provider** is a service that manages and verifies identities of users within an organization. It authenticates users and provides identity tokens that can be used to grant access to applications and services. IdPs often use standard protocols like SAML 2.0, OpenID Connect.





IAM Identity Center Permissions & Assignments

- **Multi-Account Permissions** - used to manage access across AWS accounts in our AWS Organization. We can create permission sets (collection of one or more IAM policies assigned to users and groups to define AWS access).



- **Application Assignments** - used for SSO access to many SAML 2.0 business applications. It provides required URLs, certificates and metadata.
- **Attribute-Based Acces Control (ABAC)** - fine-grained permissions based on users' attributes stored in IAM Identity Center Identity Store. Used to define permissions once, then modify AWS access by changing the attributes.

▼ ☑ **AWS Directory Services.**

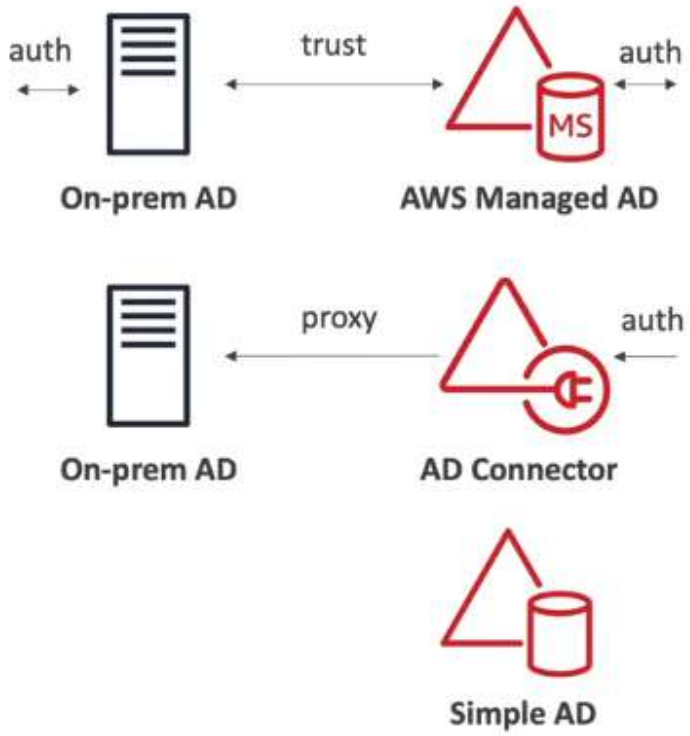
AWS Directory Service is a managed service that makes it easy to set up and run directory services in the AWS cloud or connect your AWS resources with an existing on-premises Microsoft Active Directory (AD).

Microsoft Active Directory (AD) is a directory service developed by Microsoft for Windows domain networks. It is a database of objects: user accounts, computers, printers, file shares, security groups... Objects are organized in trees and a group of trees is a forest.

AWS Directory Services

- **AWS Managed Microsoft AD** - creates our own AD in AWS, manage users locally and supports MFA. It establishers "trust" connections with our on-premise AD. Those directories are deployed across **two AZs in a region** by default and connected to our Amazon VPC.
- **AD Connector** - directory gateway (proxy) to redirect to on-premise AD, supports MFA. Users are managed on the on-premise AD.

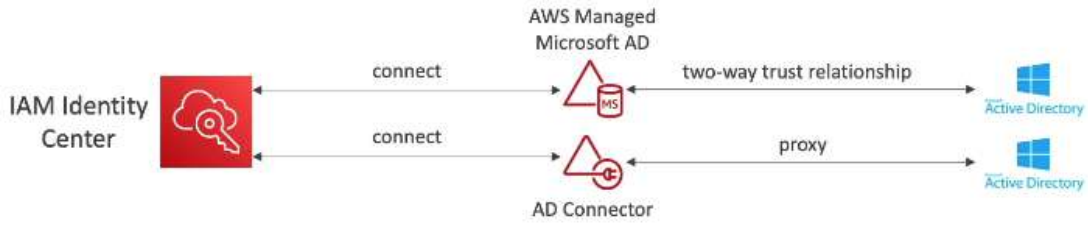
- **Simple AD** - AD-compatible managed directory on AWS. Cannot be joined with on-premise AD.



IAM Identity Center Active Directory Setup

- **Connect to an AWS Managed AD** - integration is out of the box.

- **Connect to a Self-Managed AD** - we need to create two-way trust relationship using AWS Managed Microsoft AD and AD Connector.



▼ ☑ **AWS Control Tower.**

AWS Control Tower is a service that provides a way to set up and govern a secure, multi-account AWS environment based on AWS best practices. It is designed for organizations that need to manage multiple AWS accounts and provides a pre-configured environment called a landing zone. The landing zone includes baseline IAM roles, account structures, and network configurations.

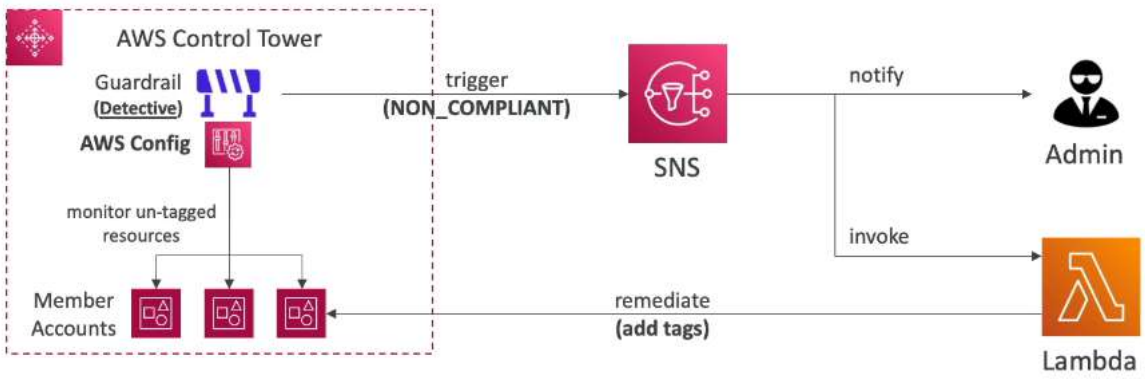
Benefits:

- Automate the set up of our environment.
- Automate ongoing policy management using guardrails.
- Detect policy violations and remediate them.
- Monitor compliance through an interactive dashboard.

Control Tower Guardrails

It provides ongoing governance for our Control Tower evironment (AWS Accounts).

- **Preventive Guardrail** - using SCPs (e.g. **restrict regions across all our accounts**).
- **Detective Guardrail** - using AWS Config (e.g. **identify untagged resources**).



Section IV

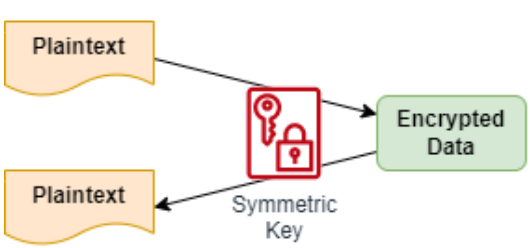
AWS Security & Encryption

▼ ☐ **KMS.**

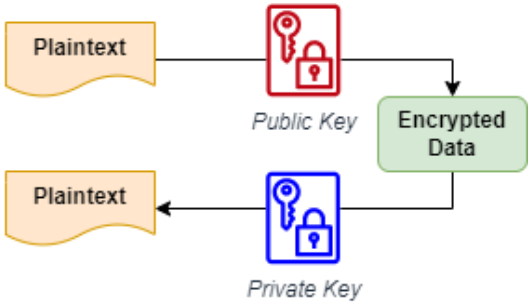
AWS KMS (Key Management Service) is a managed service that allows us to create and manage encryption keys used to encrypt and decrypt our data. It integrates with many other AWS services to simplify the encryption process.

Keys Types

- **Symmetric (AES-256 Keys)** - single encryption key that is used to encrypt and decrypt. AWS services that are integrated with KMS use Symetric CMKs. We never get access to the KMS key unencrypted (must call KMS API to use).
- **Asymmetrict (RSA & ECC Key Pairs)** - public (encrypt) and private (decrypt) key pair. Used for encrypt/decrypt or sign/verify operations. The public key is downloadable, but we cant access the private key unencrypted. It is used for encryption outside of AWS by users who cant call the KMS API.



Symmetric Encryption - the same key is used for both encrypting and decrypting data



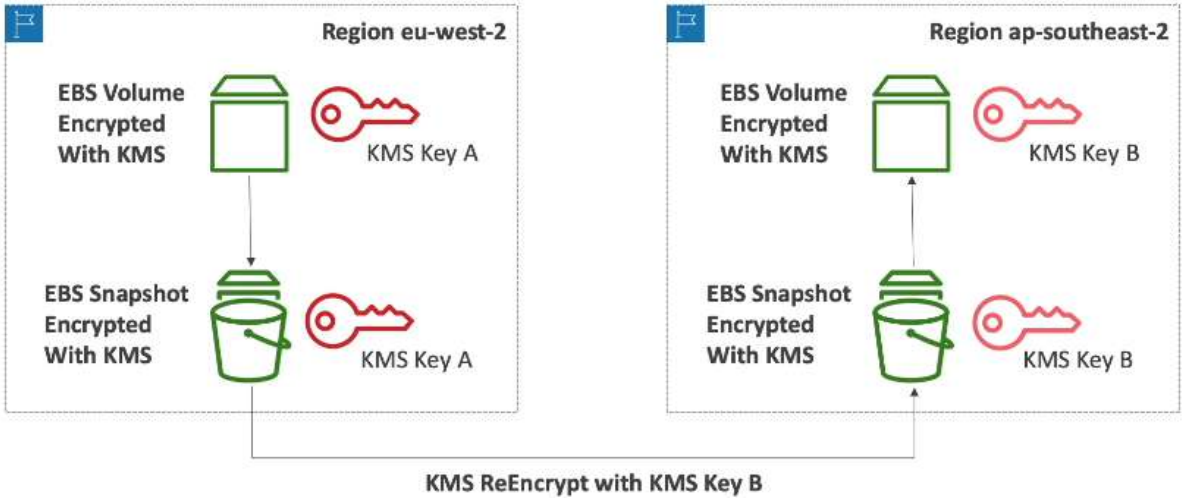
Asymmetric Encryption - the public key is used for encryption and the private key is used for decryption

Types of KMS Keys

- **Customer Managed Key (CMK)** - created, managed and used by the customer. Possibility of rotation policy (new key generated every year). Possibility to bring our own key.
- **AWS Managed Key** - created, managed and used on the customer's behalf by AWS. Used by AWS services.
- **AWS Owned Key** - collection of CMKs that an AWS service owns and manages to use in multiple accounts. AWS can use those to protect resources in our account (we cant view the keys).

Automatic key rotation is automatic every 1 year for AWS managed keys. For created CMKs it must be enabled (can be automatic or on-demand), and for imported KMS keys, only manual rotation is possible using alias.

KMS keys are scoped per region. If we want to to move encrypted EBS volume to another region we need to take snapshot of that EBS volume. Then we need to copy snapshot to another region, we neeed to re-encrypt the snapshot using a different KMS key and then we can restore EBS volume from that snapshot.



KMS Key Policies - control access to KMS keys (similar to S3 bucket policies).

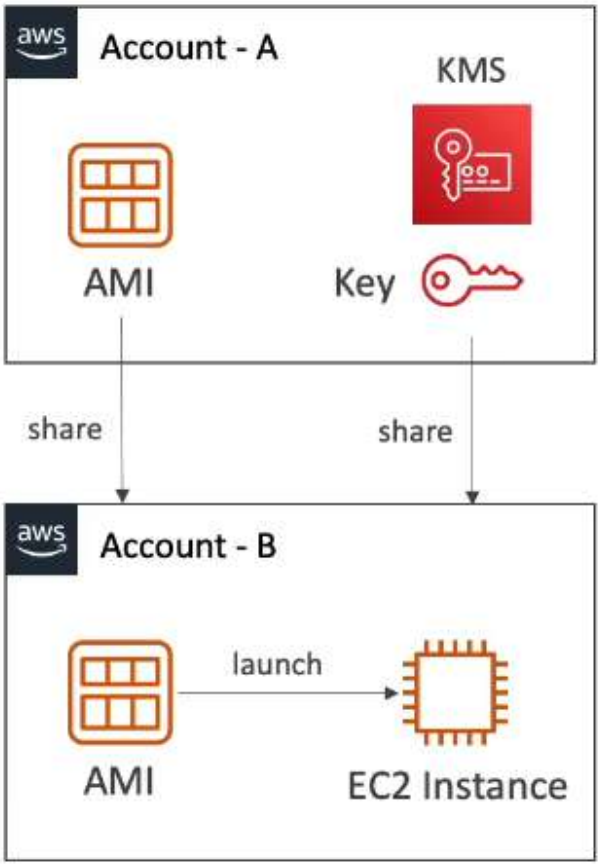
- **Default KMS Key Policy** - created if we dont provide a specific KMS Key Policy. Root user has complete access to the key.
- **Custom KMS Key Policy** - define users, roles that can access the KMS key and who can administer the key. Useful for cross-account access of our KMS key.

To copy snapshot across accounts we need to create a snapshot encrypted with our own KMS key. Then attach a KMS Key Policy to authorize cross-account access and share the encrypted snapshot. In target account we need to create a copy of the snapshot, encrypt it with CMK in our account and then create a volume from snapshot.

▼ **AMI Sharing Process Encrypted via KMS.**

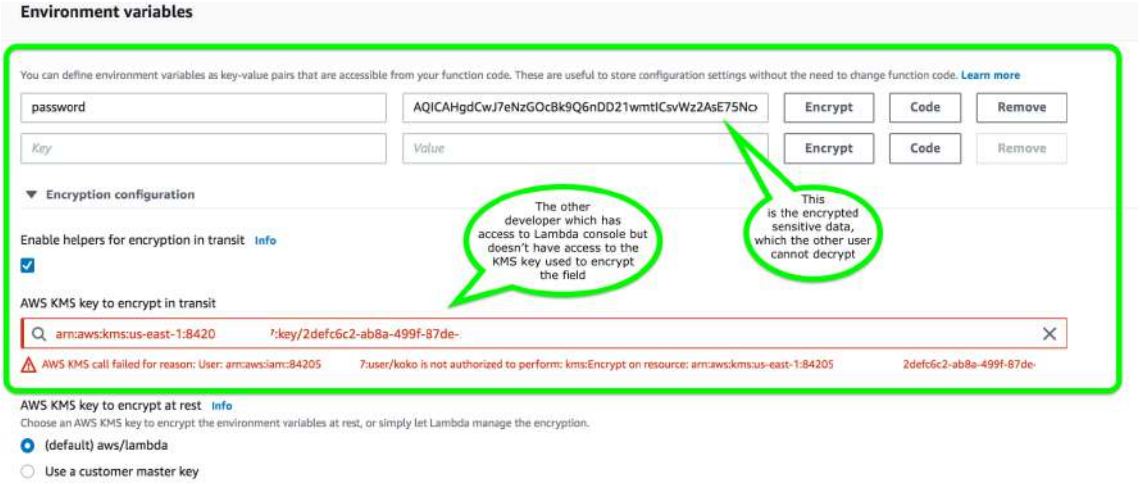
1. AMI in source account is encrypted with KMS key from source account.
2. Must modify the image attribute to add a Launch Permission which corresponds to the specified target AWS account.
3. Share the KMS keys used to encrypt the snapshot AMI references with the target account.

4. The IAM role/user in the target account must have the permissions to DescribeKey, ReEncrypt, CreateGrant, Decrypt.
5. When launching an EC2 instance from AMI, optionally the target account can specify a new KMS key in its own account to re-encrypt the volumes.



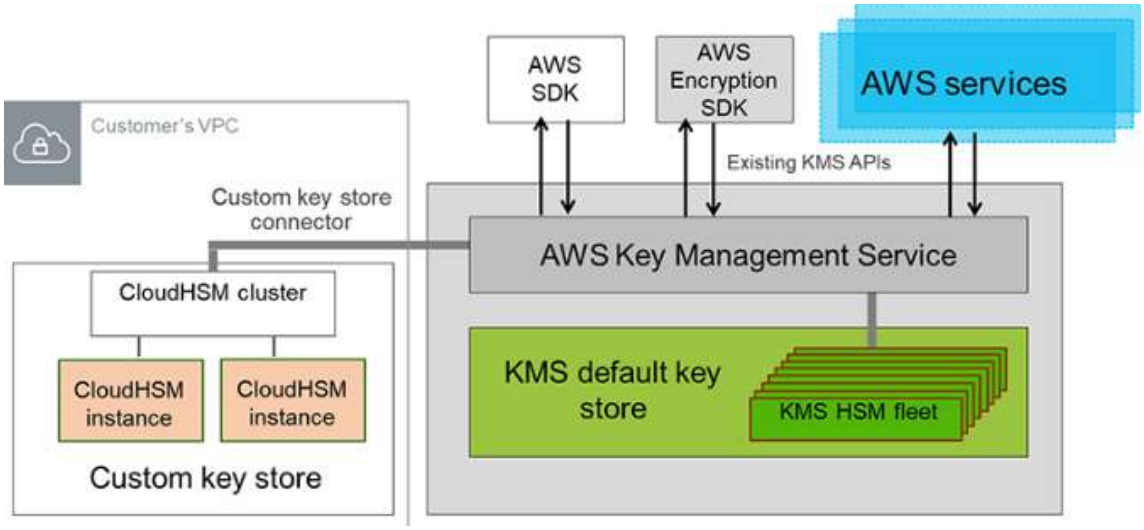
▼ **Encryption of Environment Variables in Lambda.**

Lambda encrypts the environment variables in our function by default, but the sensitive information would still be visible to other users who have access to the Lambda. The best option is to use encryption helpers to secure our environment variables.



▼ KMS - CloudHSM Integration.

The KMS custom key store feature combines the controls provided by AWS CloudHSM. We can configure our own CloudHSM cluster and authorize KMS to use it as a dedicated key store for our keys rather than the default AWS KMS key store. This is suitable if we want to be able to audit the usage of all our keys independently of KMS or CloudTrail.



▼ ☐ KMS Multi-Region Keys.

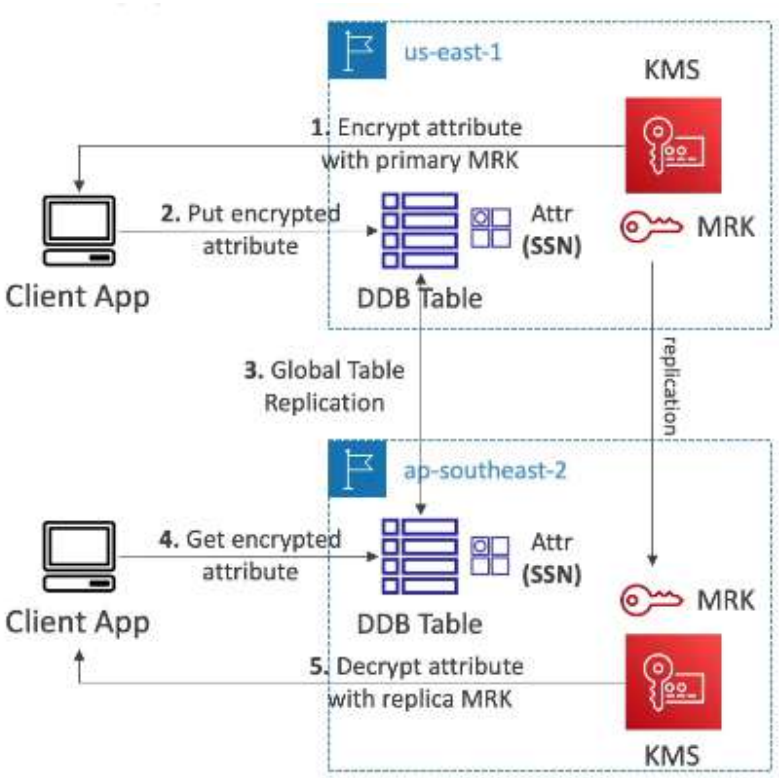
Multi-Region Keys are identical KMS keys in different regions that can be used interchangeably. They have the sam key ID, key material and automatic rotation. We can encrypt in one region and decrypt in other regions (no need to re-encrypt or make cross-region API calls).

KMS Multi-Region is not global (primary + replicas). Each Multi-Region key is managed idependently.

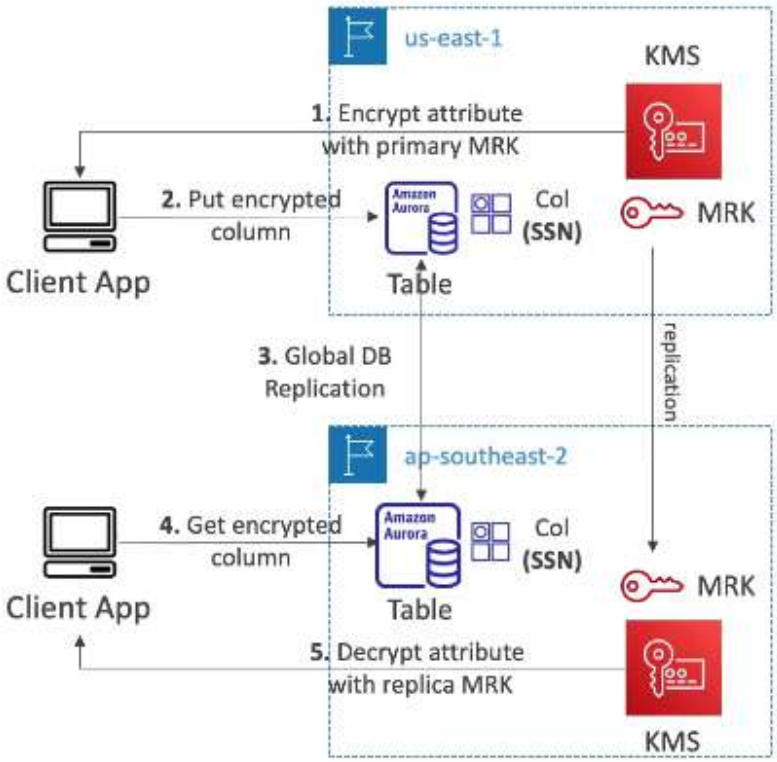
DynamoDB Global Tables/Global Aurora & KMS MRK Client-Side Encryption

We can encrypt specific attributes client-side in our DynamoDB table using DynamoDB Encryption Client. Combined with Global Tables, the client-side encrypted data is replicated to other regions. If we use a MRK replicated in the same region as the DynamoDB Global table, then clients in these regions can use low-latency API calls to KMS in their region to decrypt the data client-side.

Using client-side encryption we can protect specific fields and guarantee only decryption if the client has access to an API key.



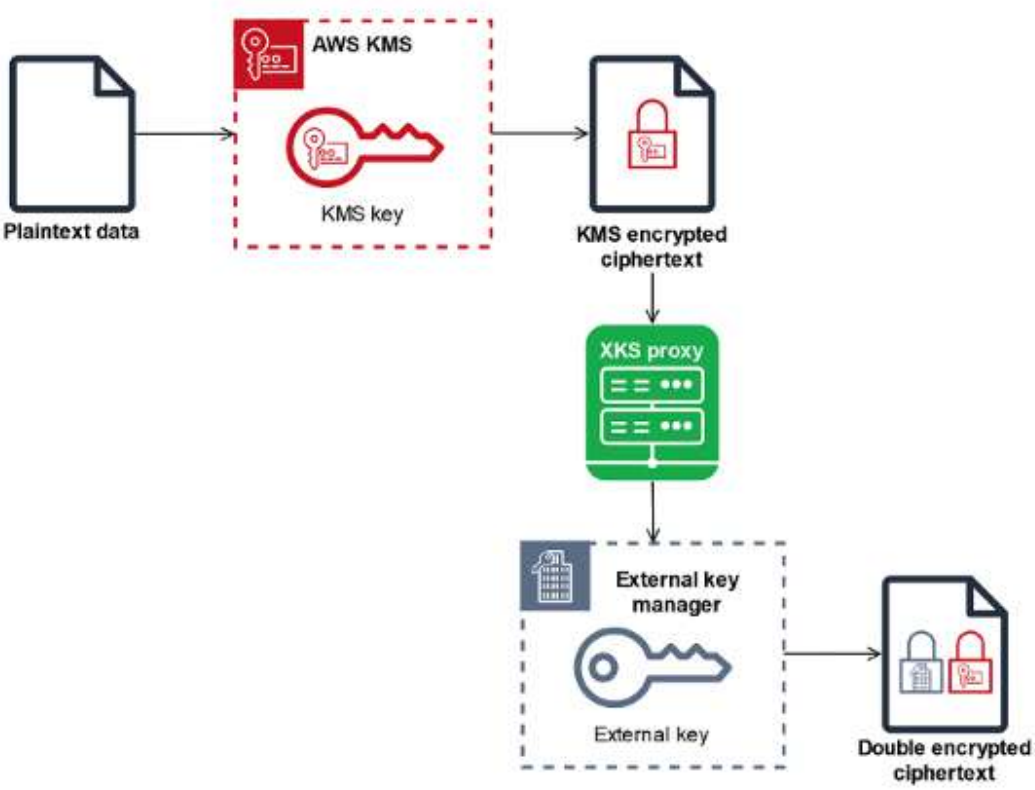
We can encrypt specific attributes client-side in our Aurora table using the AWS Encryption SDK. Combined with Aurora Global Tables, the client-side encrypted data is replicated to other regions. We can protect specific fields even from database admins.



▼ ☐ **KMS XKS.**

External Key Store (XKS) in KMS is a setup where encryption keys are stored and managed outside the primary KMS infrastructure. It enhances security and compliance by allowing organizations to retain control over their encryption keys in a dedicated external system.

XKS Proxy is an intermediary service that acts as a bridge, enabling secure key retrieval and management without requiring direct integration with the external key store itself.



▼ ☑ **S3 Replication with Encryption.**

Unencrypted objects and objects encrypted with SSE-S3 are replicated by default. Object encrypted with SSE-C can also be replicated.

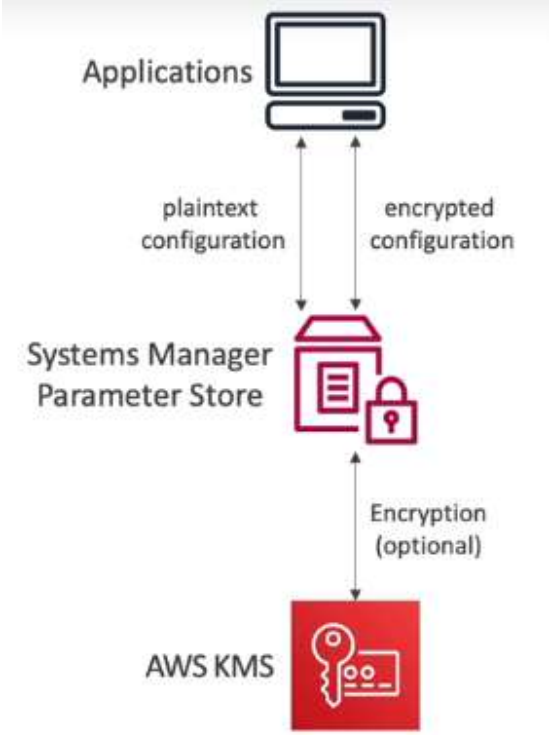
For objects encrypted with SSE-KMS, we need to enable the option:

1. Specify which KMS Key to encrypt the objects within the target bucket.
2. Adapt the KMS Key Policy for the target key.
3. Create IAM Role with kms:Decrypt for the source KMS Key and kms:Encrypt for the target KMS Key.

We can use MRK, but they are currently treated as independent keys by S3 (**the object will still be decrypted and then encrypted**).

▼ ☑ **SSM Parameter Store.**

SSM Parameter Store - it is a secure storage for configuration and secrets (API keys, passwords, configurations ...). It is serverless, scalable and durable. Also it is secure because we control access permissions using IAM.



Parameters Policies - allow us to assign a TTL to a parameter to force updating or deleting sensitive data such as passwords.

Expiration (to delete a parameter)

```
{
    "Type":"Expiration",
    "Version":"1.0",
    "Attributes":{
        "Timestamp":"2020-12-02T21:34:33.000Z"
    }
}
```

ExpirationNotification (EventBridge)

```
{
    "Type":"ExpirationNotification",
    "Version":"1.0",
    "Attributes":{
        "Before":"15",
        "Unit":"Days"
    }
}
```

NoChangeNotification (EventBridge)

```
{
    "Type":"NoChangeNotification",
    "Version":"1.0",
    "Attributes":{
        "After":"20",
        "Unit":"Days"
    }
}
```

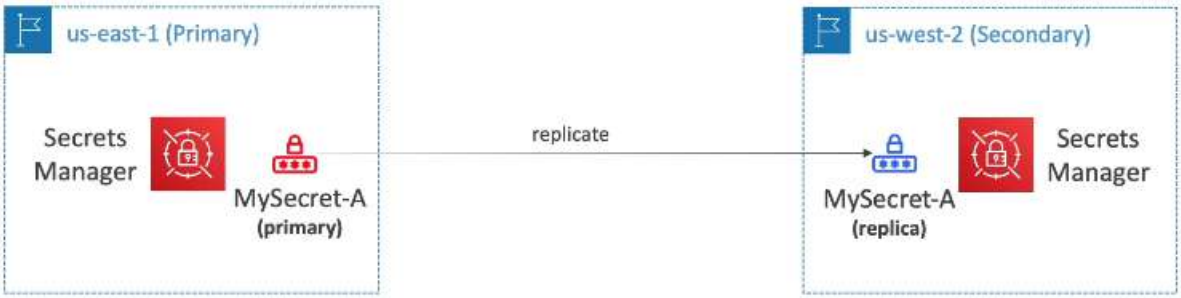
▼ ☑ **AWS Secrets Manager.**

ASM is a newer service which is used for storing secrets. It has capability to force rotation of secrets every X days. We can automate generation of secrets on rotation using Lambda. All secrets are encrypted using KMS.

Can be integrated with Amazon RDS (MySQL, Postgres, Aurora). Mostly meant for RDS integration.

We can replicate secrets across multiple regions. Secrets Manager keeps read replicas in sync with the primary secret. There is an ability to promote a read replica secret to a standalone secret.

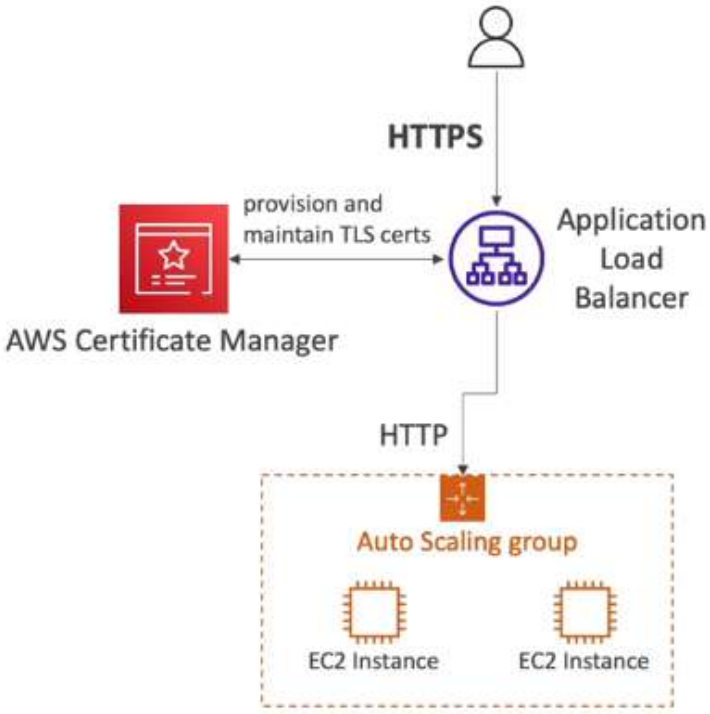
Use cases: multi-region apps, disaster recovery strategies, multi-region databases...



▼ ☑ **AWS ACM & IAM Certificate Store.**

ACM lets us easily provision, manage and deploy TLS certificates. Used to provide HTTPS for websites. It supports both public and private TLS certificates.

Can be integrated with (load TLS certificates on) ELB, CloudFront distributions, APIs on API Gateway.



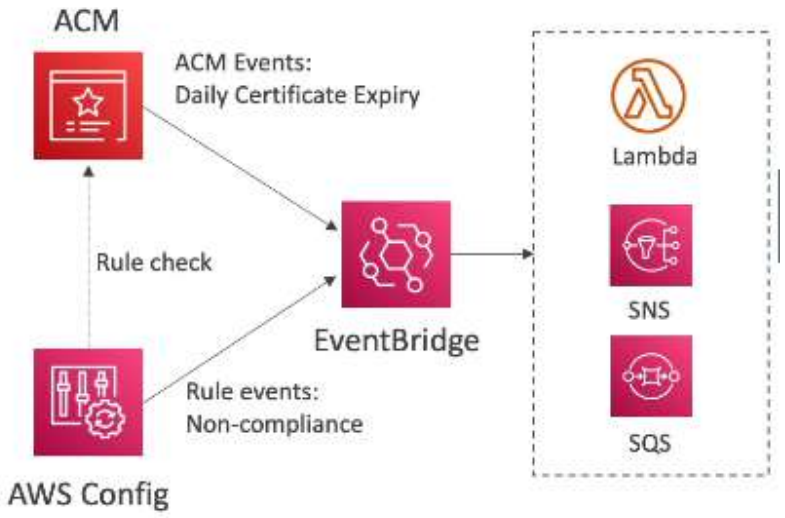
ACM Private CA - for enterprise customers building a public key infrastructure (PKI) inside the AWS cloud and is intended for private use within an organization. We can create our own CA hierarchy and issue certificates with it for

authenticating internal users, computers, applications, services... Certificates issued by a private CA are trusted only within our organization, not on the internet.

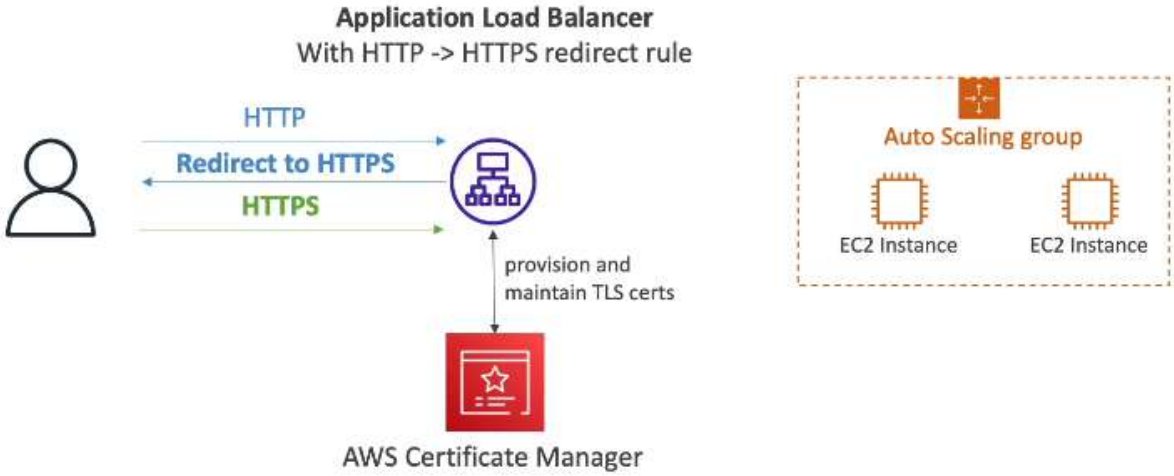
Importing Public Certificates

There is an option to generate the certificate outside of ACM and then import it. Because there is **no automatic renewal**, we must import a new certificate before expiry. ACM sends daily expiration events, starting 45 days prior to expiration (number of days can be configured). Those events are appearing in EventBridge.

AWS Config has a managed rule named *acm-certificate-expiration-check* to check for expired certificates.



Integration with ALB

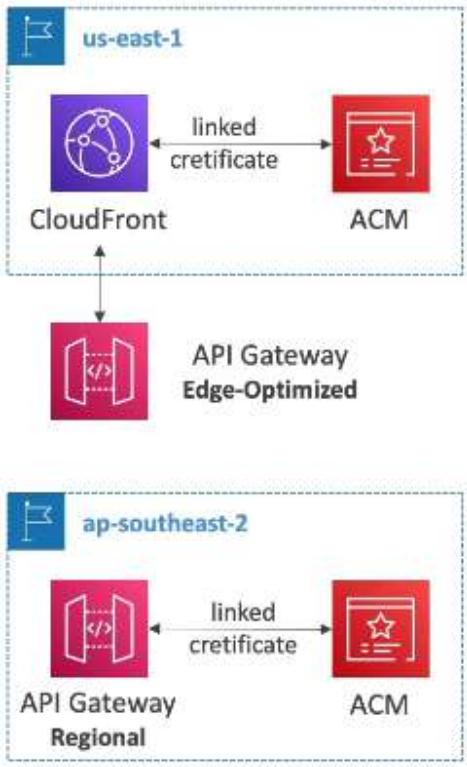


Integration with API Gateway

First, we need to create a custom domain name in API Gateway. ACM can be integrated with Edge-Optimized and Regional endpoints in API Gateway.

- Edge-Optimized Endpoint (default) - TLS Certificate must be in the same region as CloudFront.
- Regional Endpoint - TLS Certificate must be imported on API Gateway in the same region as the API Stage.

Then we need to setup CNAME or Alias record in Route 53.



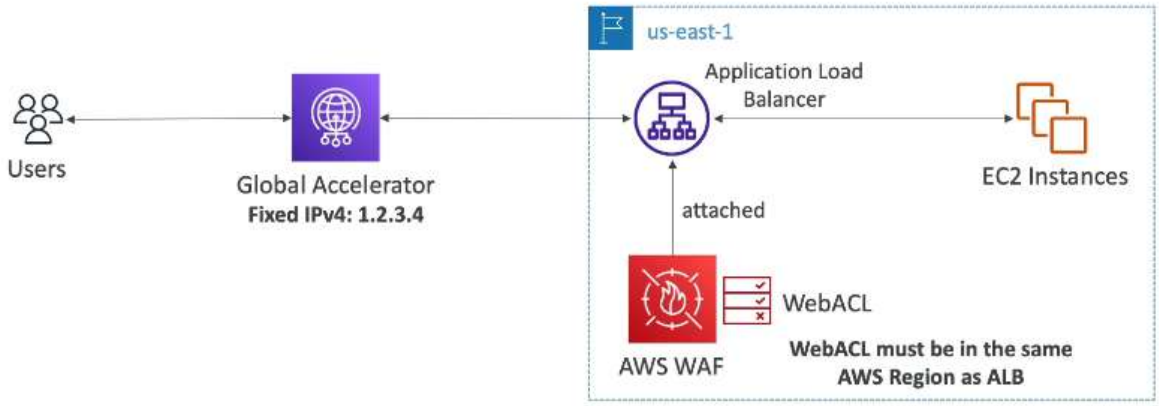
IAM Certificate Store is a system or service used to manage and organize digital certificates within an IAM framework.

▼ ☑ **AWS WAF.**

AWS WAF (Web Application Firewall) - protects our web applications from common web exploits (**Layer 7**). It can be deployed on **ALB, API Gateway, CloudFront, AppSync** and **Cognito User Pool**.

On WAF we can define **Web ACL** (Web Access Control List). Rules can include IP addresses, HTTP headers, HTTP body or URI strings. It can protect us from SQL Injection and XSS. There are Rate-based rules which are used for DDoS protection. A **rule group** is a reusable set of rules that we can add to a web ACL. Web ACL are regional except for CloudFront.

If we want to get a fixed IP in our application while using WAF with an ALB, we have to use Global Accelerator to get fixed IP for application and then enable WAF on our ALB.

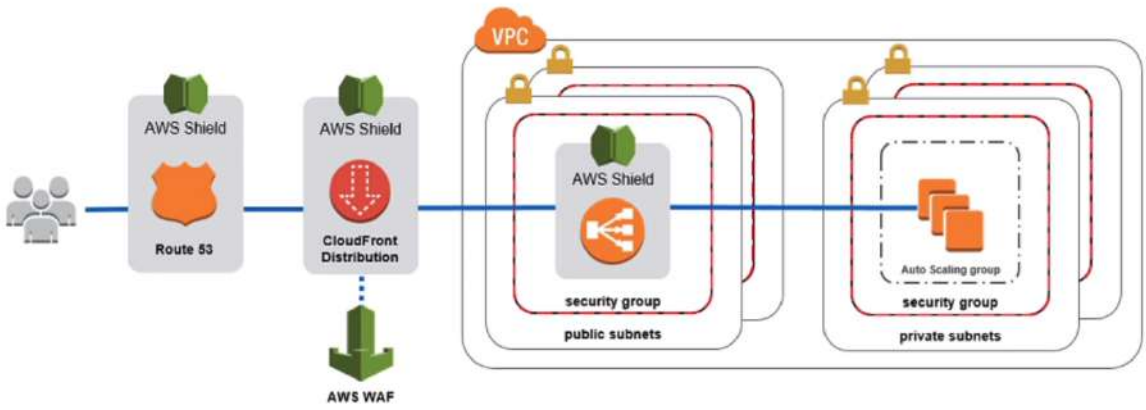


▼ ☑ **AWS Shield.**

DDoS Protection on AWS:

- **AWS Shield Standard** - protects against DDoS attack for our website and applications, its free for all customers.
- **AWS Shield Advanced** - 24/7 premium DDoS protection ($3000 per month).
- **CloudFront & Route 53** - combined with AWS Shield, provides attack mitigation at the edge.

We should be ready to scale during attack - leverage AWS Auto Scaling.



▼ ☑ **Firewall Manager.**

AWS Firewall Manager manages security rules in all accounts of an AWS Organization. Rules are applied to new resources as they are created (good for compliance) across all and future accounts in our organization.

Common set of security rules: VPC Security Groups for EC2, WAF rules, AWS Shield Advanced, AWS Network Firewall ...

▼ ☑ **Amazon GuardDuty.**

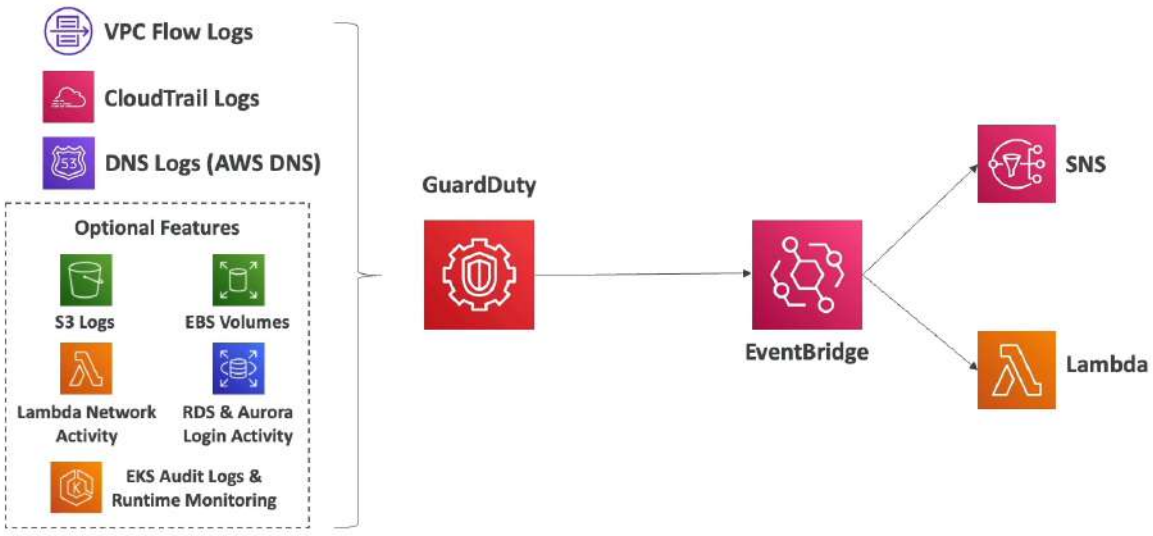
GuardDuty is an intelligent threat discovery service which protects our AWS account. It uses ML algorithms, anomaly detection and 3rd party data. Focuses on real-time threat detection.

Input data can include:

- **CloudTrail Event Logs** - unusual API calls, unauthorized deployments.
- **VPC Flow Logs** - unusal internal traffic, unusual IP address.
- **DNS Logs** - compromised EC2 instances sending encoded data within DNS queries.
- **Optional Features** - EKS Audit Logs, RDS & Aurora, EBS, Lambda, S3 Data Events...

We can setup EventBridge rules to be notified in case of findings. EventBridge rules can target AWS Lambda or SNS.

GuardDuty can protect us against CryptoCurrency attacks.



▼ ☑ **Amazon Inspector.**

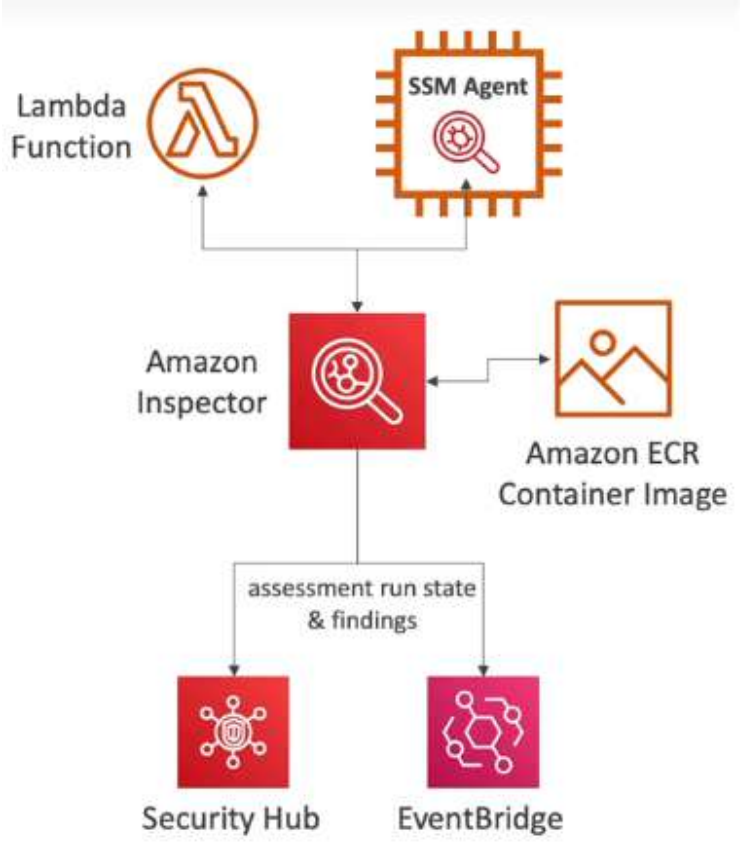
Amazon Inspector is a security assessment service that helps improve the security and compliance of applications deployed on AWS. Inspector

automatically assesses applications for vulnerabilities or deviations from best practices (CVE).

Can be used for:

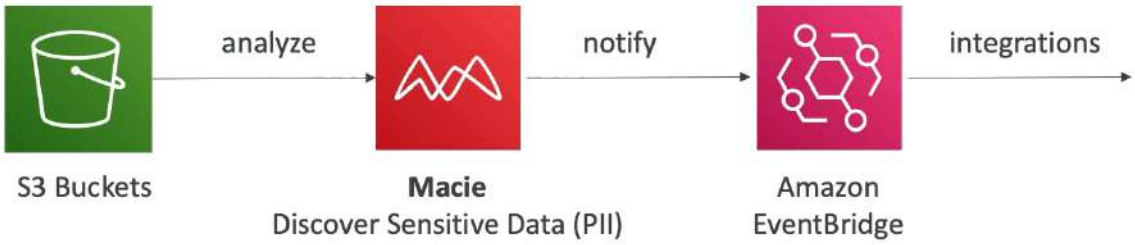
- EC2 Instances
  - Leveraging the SSM agent.
  - Analyze against unintended network accessibility.
  - Analyze the running OS against known vulerabilities.
- Container images push to Amazon ECR
  - Assessment of container images as they are pushed.
- Lambda functions
  - Identifies software vulnerabilities in function code and package dependencies.
  - Assessment of funtions as they are deployed.

Can be integrated with AWS Security Hub and send findings to EventBridge.



▼ ☑ **Amazon Macie.**

Amazon Macie is a fully managed data security and data privacy service that uses ML and pattern matching to discover and protect our sensitive data in AWS. Macie helps idenfity and alert us to sensitive data in our **S3 Buckets**, such as personally identifiable information (PII).



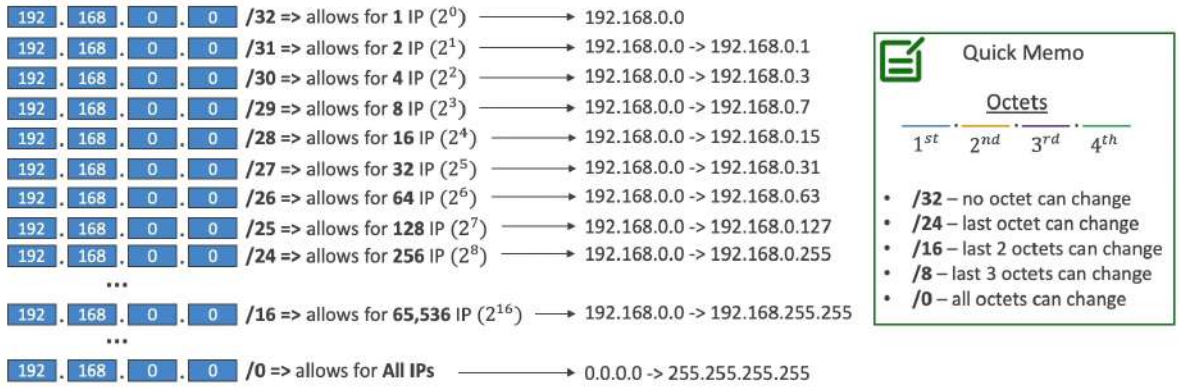
Networking - VPC

▼ ☑ **CIDR, Private & Public IP.**

CIRD (Classless Inter'Domain Routing) - a method for allocationg IP addresses. It is used in Security Groups rules and AWS networking in general. It help to define an IP address range.

CIDR consists of two components:

- **Base IP** - represents an IP contained in the range.
- **Subnet Mask** - defines how many bits can change in the IP.



Private IP addresses are used within a VPC and are not routable over the internet. They are used for internal communication between resources within the same VPC or between different VPCs if they are peered or connected via a VPN.

Public IP addresses are routable over the internet. They are used to allow resources to communicate with external networks and to be accessible from the internet.

▼ ☑ VPC & Subnet.

VPC (Virtual Private Cloud) is a private network where we can deploy our resrouces (regional resource). All new AWS accounts have a default VPC. New EC2 instances are launched into the default VPC if no subnet is specified. Default VPC has internet connectivity and all EC2 instances inside it have public IPv4 addresses.

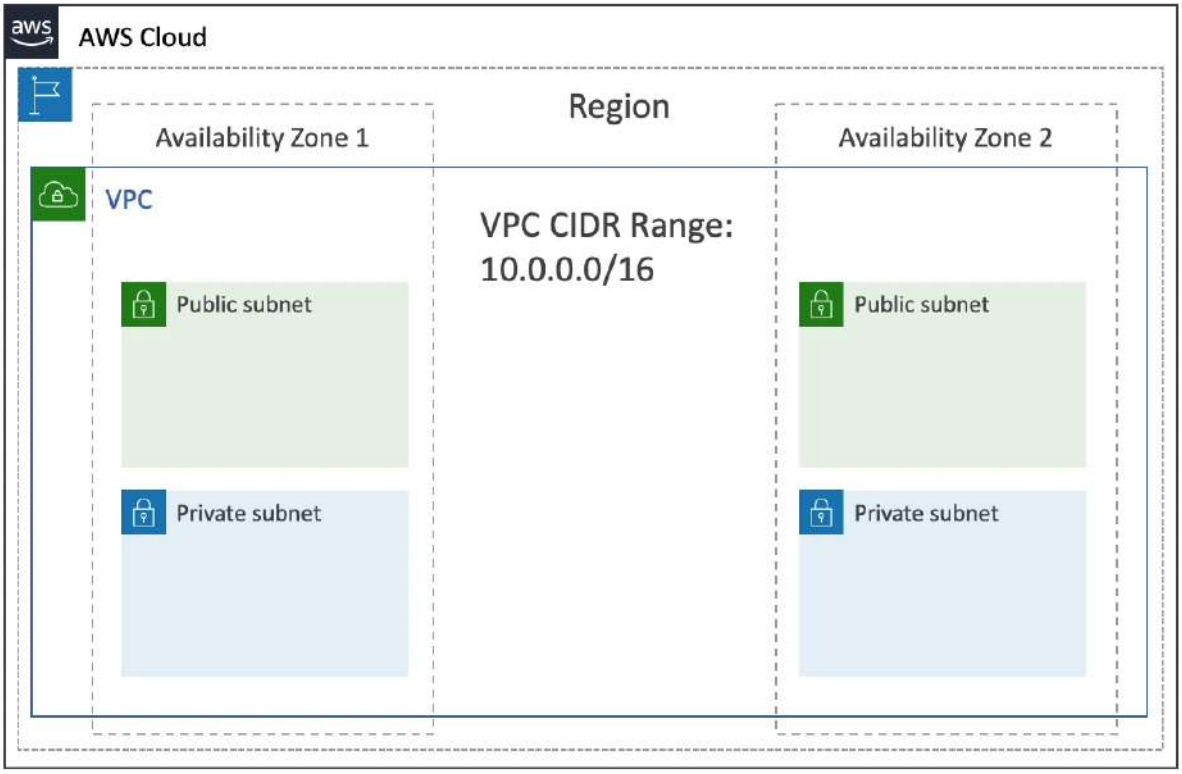
Max CIDR per VPC is five. For each CIDR:

- Min size is /28 (16 IP addresses)
- Max size is /16 (65536 IP addresses)

Our VPC CIDR should not overlap with our other networks (e.g. corporate).

Subnets allow us to partition our network inside our VPC (AZ resource).

- Public subnet - subnet that is accessible from the internet.
- Private subnet - subnet that is not accesible from internet



AWS reserves 5 IP addresses (first 4 and last 1) in each subnet. These IP addresses are not available for use and cant be assigned to an EC2 instance.

Example: If CIDR block is 10.0.0.0/24, then reserved IP addresses are:

10.0.0.0	Network Address
10.0.0.1	For the VPC router
10.0.0.2	For mapping to Amazon-provided DNS
10.0.0.3	For future use
10.0.0.255	Network Broadcast (AWS does not support broadcast in VPC)

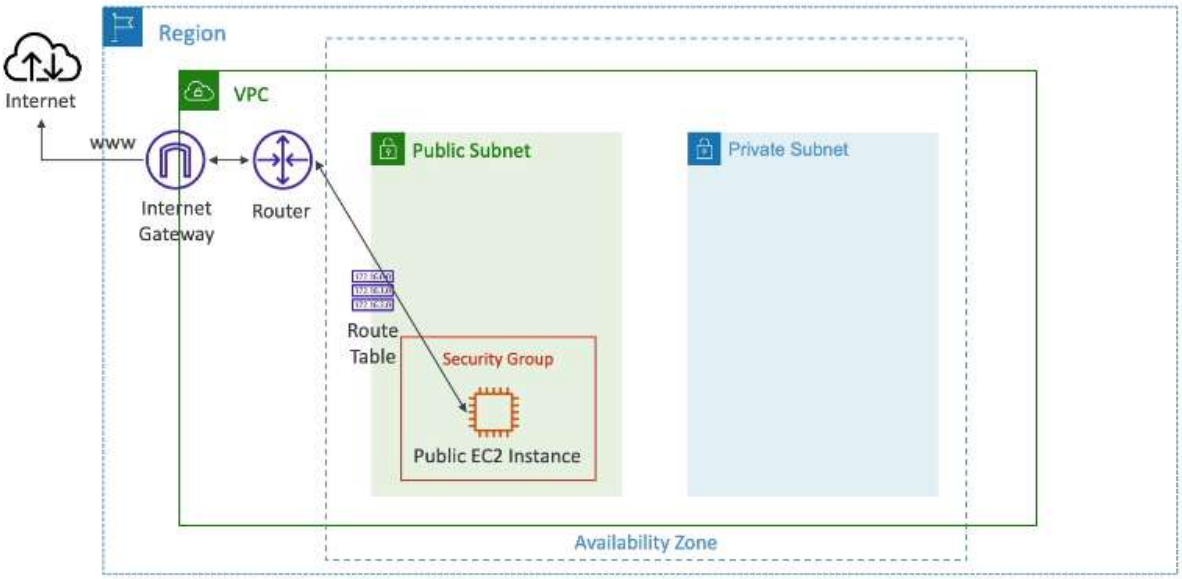
▼ ☑ **Internet Gateways & Route Tables.**

Internet Gateway is a horizontally scaled, redundant, and highly available VPC component that allows instances within our VPC to connect to the internet, and

for the internet to initiate connections with instances within our VPC (if allowed by security rules).

One VPC can only be attached to one IGW and vice versa.

Routing Table is a set of rules (routes) that determine where network traffic is directed. Every subnet in our VPC must be associated with a route table.

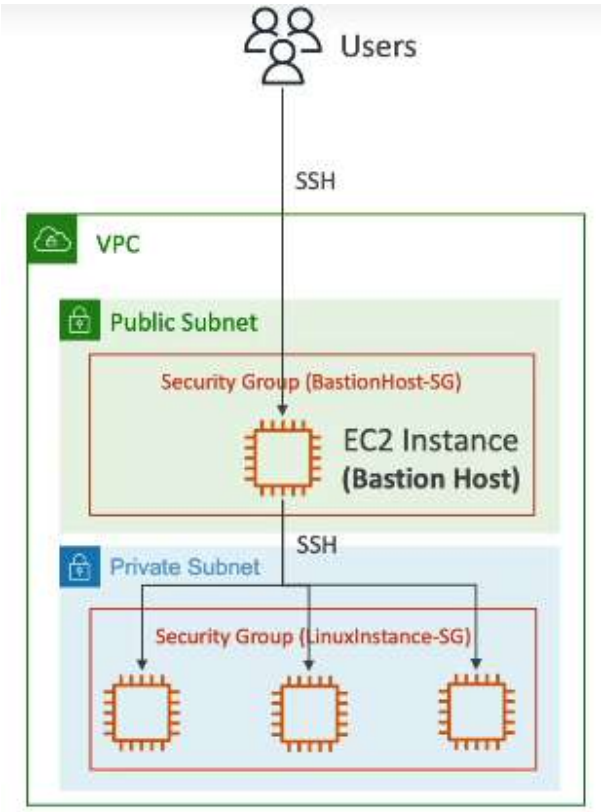


▼ ☑ **Bastion Hosts.**

Bastion Host is a special-purpose instance designed to provide secure access to our private network in the AWS cloud. It acts as a bridge or "**jump box**" for administrative tasks, allowing us to securely manage and administer our instances in private subnets.

Bastion Host security group must allow inbound from the internet on port 22 from restricted CIDR, for example the public CIDR of our corporation.

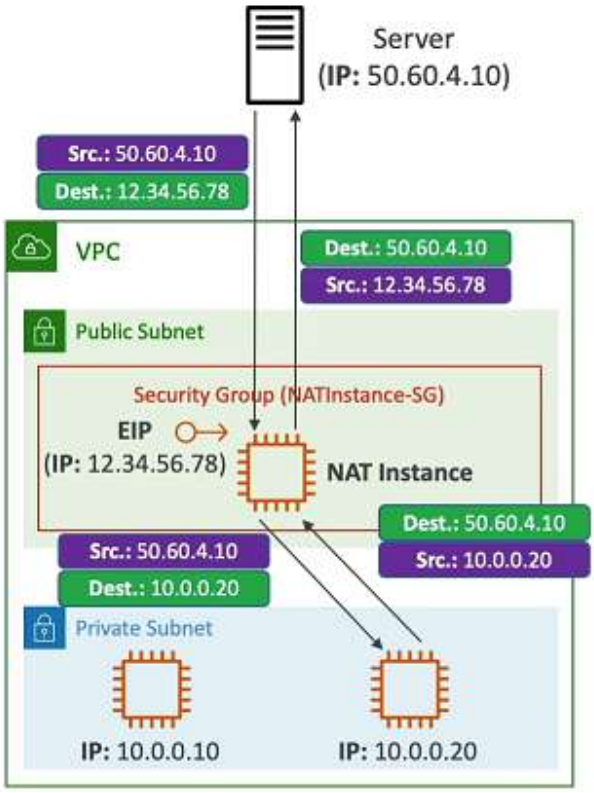
Security Group of the EC2 instances must allow the Security Group of the Bastion Host, or the private IP of the Bastion Host.



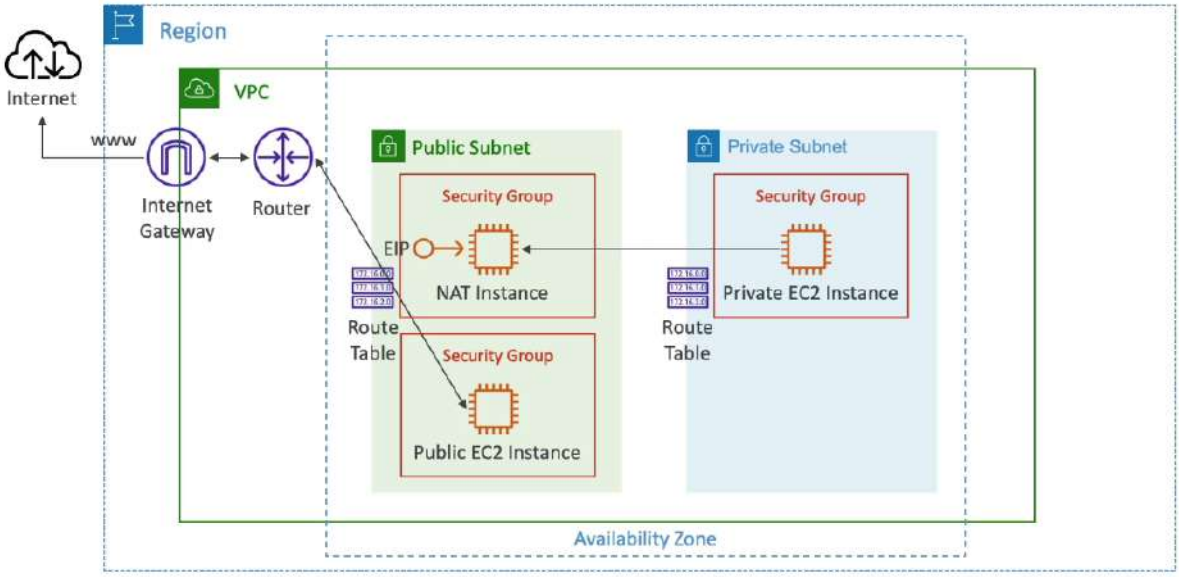
▼ ☑ **NAT Instances & NAT Gateways.**

NAT Instance allows EC2 instance in private subnets to connect to the internet. They must be launched in a public subnet and we must disable EC2 source/destination check setting and have Elastic IP attached to it.

Route Tables must be configured to route traffic from private subnets to the NAT Instance.

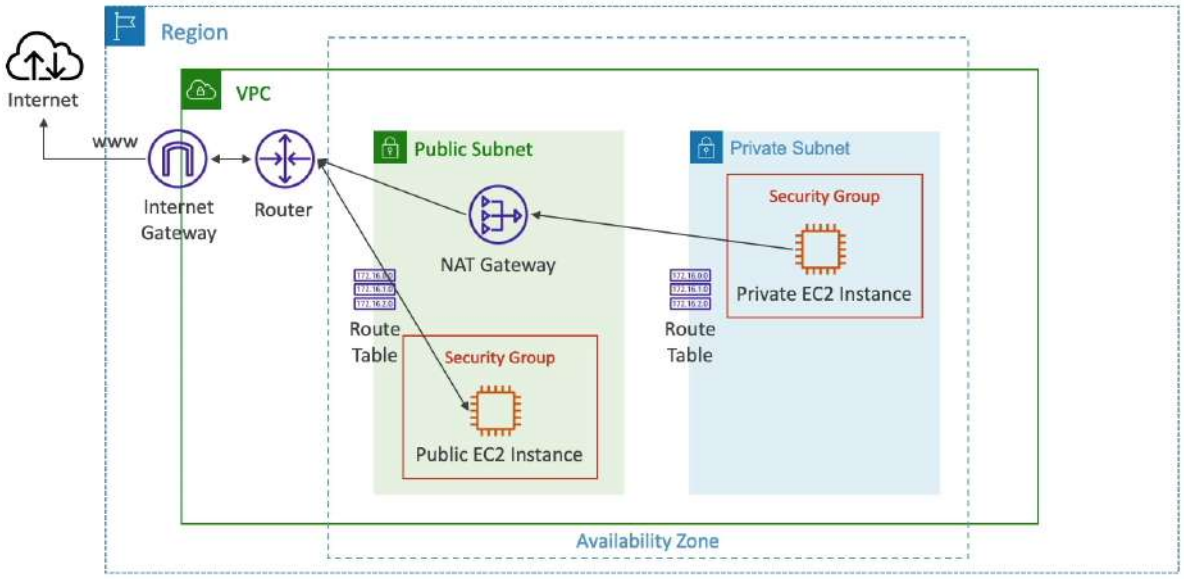


There is a pre-configured Amazon Linux AMI for NAT Instance. NAT instances are not highly available, we need to create an ASG in multi-AZ and internet traffic bandwidth depends on EC2 instance type.

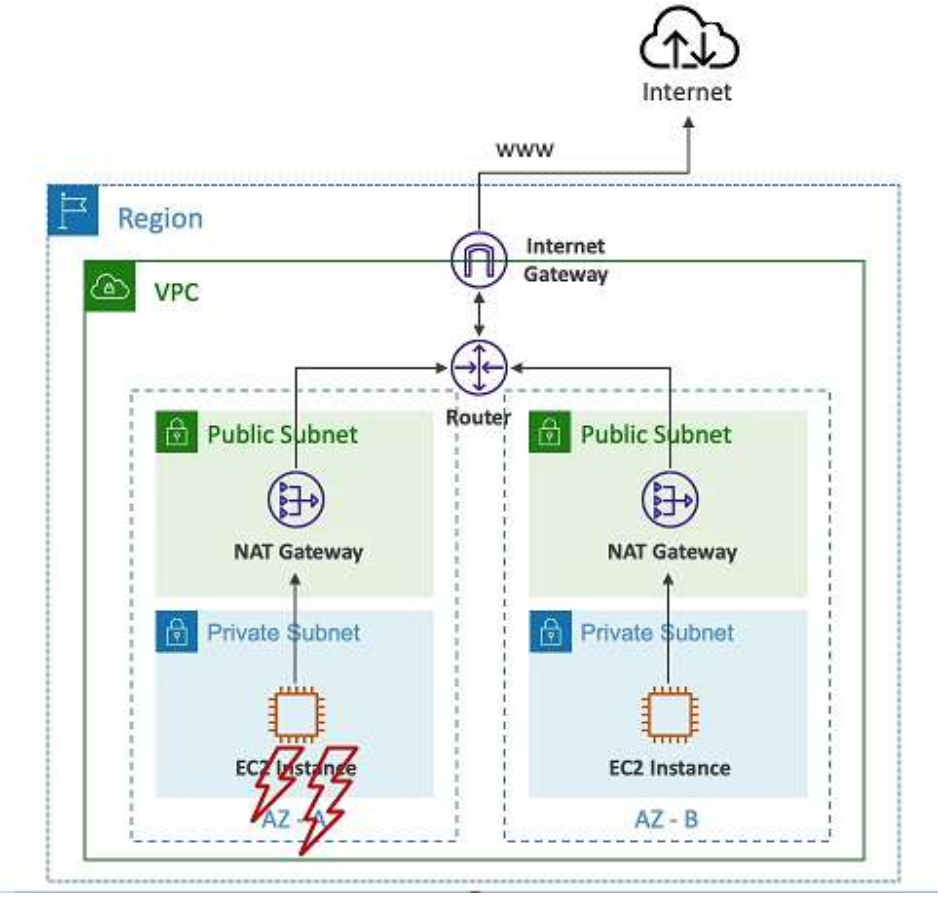


NAT Gateway is AWS managed NAT with higher bandwidth, high availability and no administration. We pay per hour for usage and bandwidth. NAT Gateway is created in a specific AZ and uses an Elastic IP.

It cant be used by EC2 instance in the same subnet (only from other subnets). It requires an Internet Gateway.



NAT Gateway is resilient within a single AZ. We must create multiple NAT Gateways in multiple AZs for fault-tolerance. There is no cross-AZ failover needed because if an AZ goes down it doesnt need NAT.



NAT Instance vs NAT Gateway

	NAT Gateway	NAT Instance
Availability	Highly available within AZ (create in another AZ)	Use a script to manage failover between instances
Bandwidth	Up to 100 Gbps	Depends on EC2 instance type
Maintenance	Managed by AWS	Managed by you (e.g., software, OS patches, ...)
Cost	Per hour & amount of data transferred	Per hour, EC2 instance type and size, + network $
Public IPv4	✓	✓
Private IPv4	✓	✓
Security Groups	✗	✓
Use as Bastion Host?	✗	✓

▼ ☑ **NACL & Security Groups.**

NACL - firewall which controls traffic from and to subnet. It can have ALLOW and DENY rules. Rules only include IP addresses. Newly created NACLs will deny everything. We can have one NACL per subnet, new subnets are assigned the Default NACL.

Default NACL accepts everything inbound/outbound with the subnets it is associated with. We should not modify the Default NACL, instead we should create custom NACLs.

Default NACL for a VPC that supports IPv4

Inbound Rules

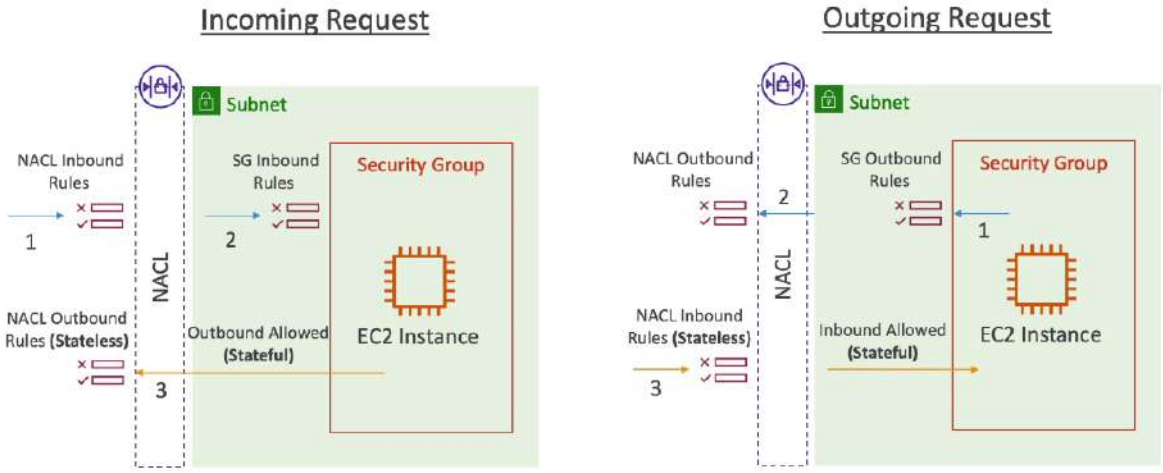
Rule #	Type	Protocol	Port Range	Source	Allow/Deny
100	All IPv4 Traffic	All	All	0.0.0.0/0	ALLOW
*	All IPv4 Traffic	All	All	0.0.0.0/0	DENY

Outbound Rules

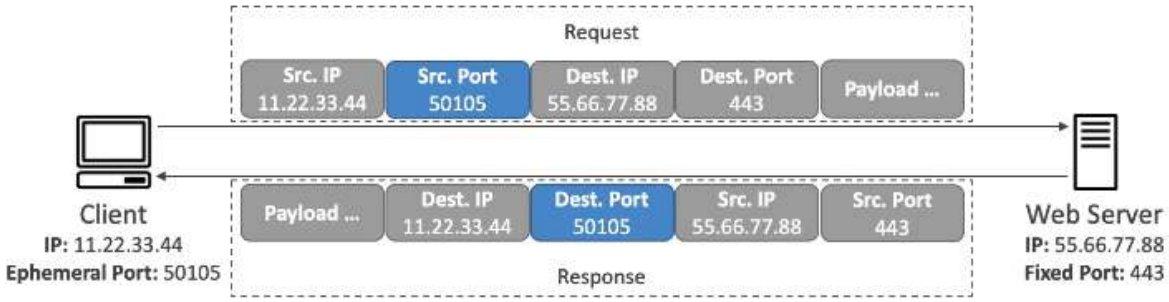
Rule #	Type	Protocol	Port Range	Destination	Allow/Deny
100	All IPv4 Traffic	All	All	0.0.0.0/0	ALLOW
*	All IPv4 Traffic	All	All	0.0.0.0/0	DENY

Rules have a number (1-32766), higher precedence = lower number. First rule match will drive the decision.

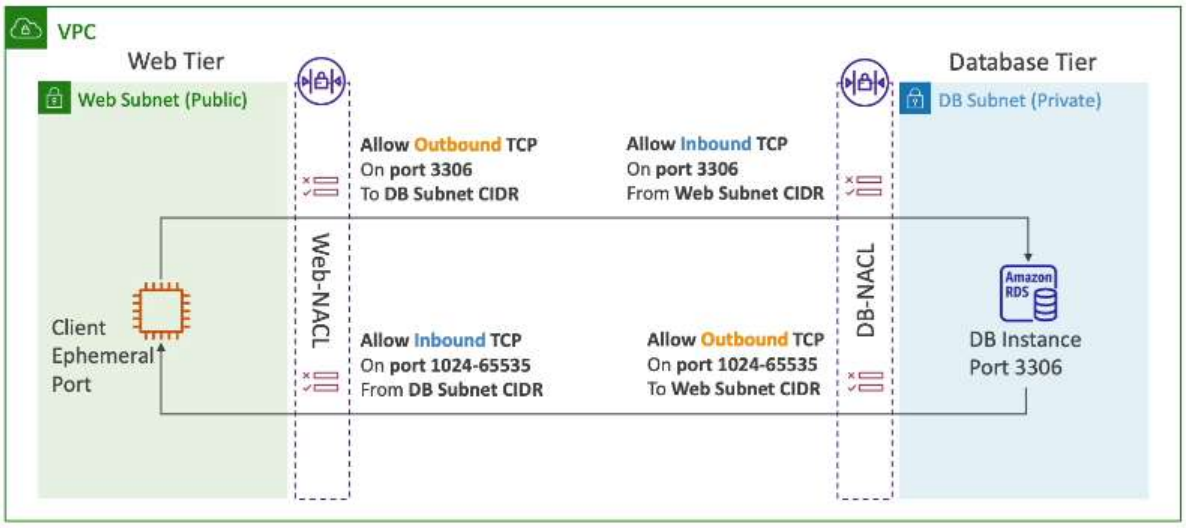
Security Groups - firewall that controls traffic to and from an EC2 instance. It can have only ALLOW rules. Rules include IP addresses and other security groups.



Ephemeral Ports - temporary, short-lived ports used for outbound connections. They are typically assigned dynamically and used for communication between a client and a server.



NACL with Ephemeral Ports



- **Client → Web Server:** The client connects to the web server using an ephemeral port.
- **Web Server → Database:** The web server connects to the RDS instance over port 3306 to retrieve or store data.
- **Response from Database → Web Server:** The RDS instance sends responses back to the web server using an ephemeral port.
- **Response from Web Server → Client:** The web server sends the response back to the client.

NACL rules ensure that the web tier can communicate with the database tier while limiting traffic to only the necessary ports. The use of a private subnet for the database ensures it is not directly accessible from the internet, enhancing security.

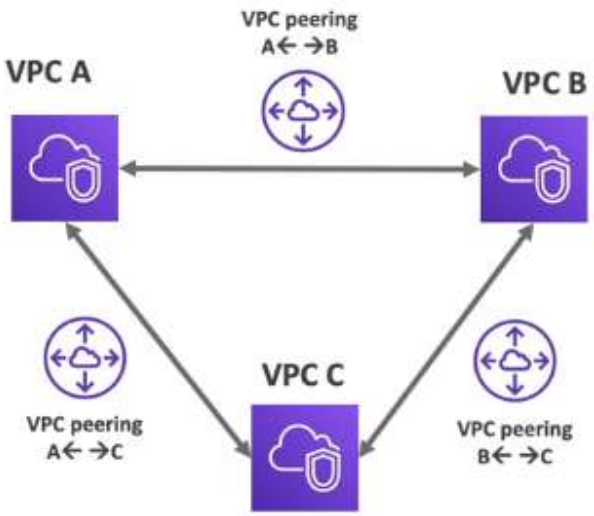
Security Group vs NACL

Security Group	NACL
Operates at the instance level	Operates at the subnet level
Supports allow rules only	Supports allow rules and deny rules
Stateful: return traffic is automatically allowed, regardless of any rules	**Stateless:** return traffic must be explicitly allowed by rules (think of ephemeral ports)
All rules are evaluated before deciding whether to allow traffic	Rules are evaluated in order (lowest to highest) when deciding whether to allow traffic, first match wins
Applies to an EC2 instance when specified by someone	Automatically applies to all EC2 instances in the subnet that it's associated with

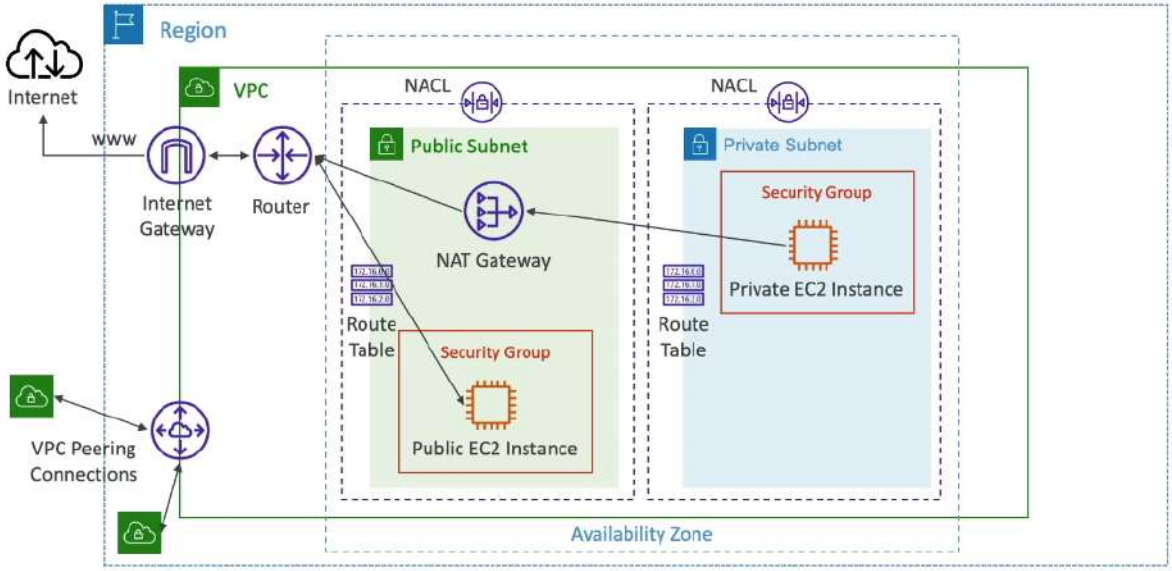
▼ ☑ **VPC Peering.**

VPC Peering connects two VPCs privately using AWS network. It makes two VPCs behave as if they were in the same network. They must not have overlapping CIDR (IP address range).

VPC Peering connection is not transitive (must be established for each VPC that need to communicate with one another).

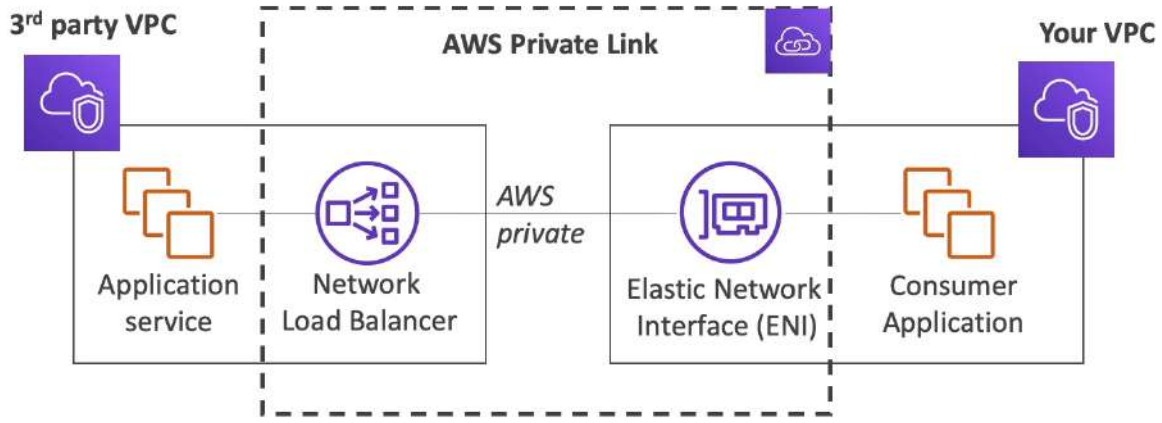


We must update route tables in each VPC's subnets to ensure EC2 instances can communicate with each other. We can create VPC Peering connection between VPCs in different AWS accounts/regions. Also we can reference a security group in a peered VPC (works cross accounts in same region).



▼ ☑ **VPC Endpoints & AWS PrivateLink.**

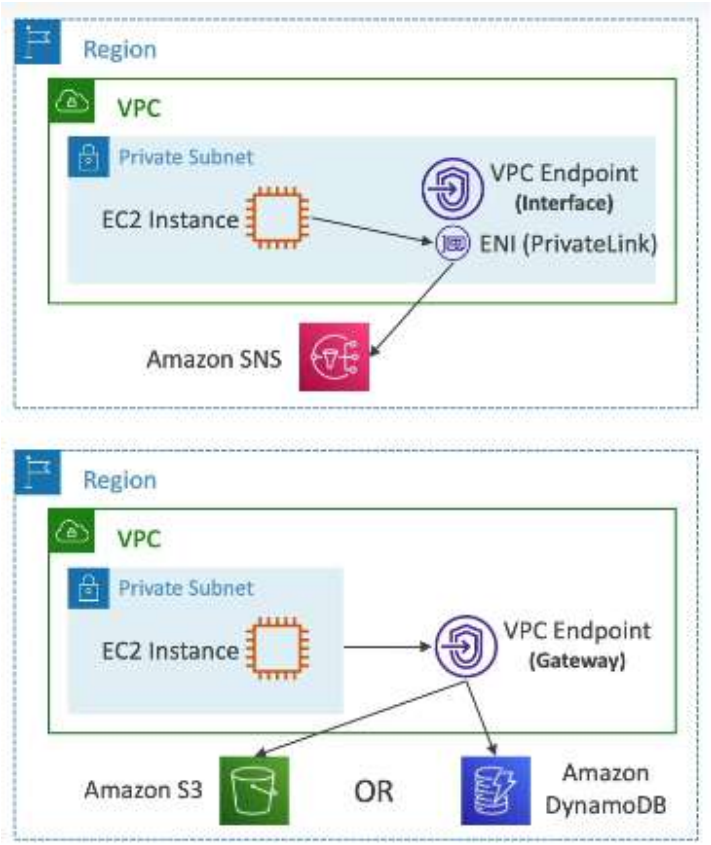
AWS PrivateLink allows us to privately access services hosted on AWS, including our own applications, third-party services, and AWS services, without exposing our traffic to the public internet. It simplifies the network architecture, improves security. Does not require VPC peering, internet gateway, NAT, route tables, but it requires a NLB on service VPC and ENI on customer VPC.

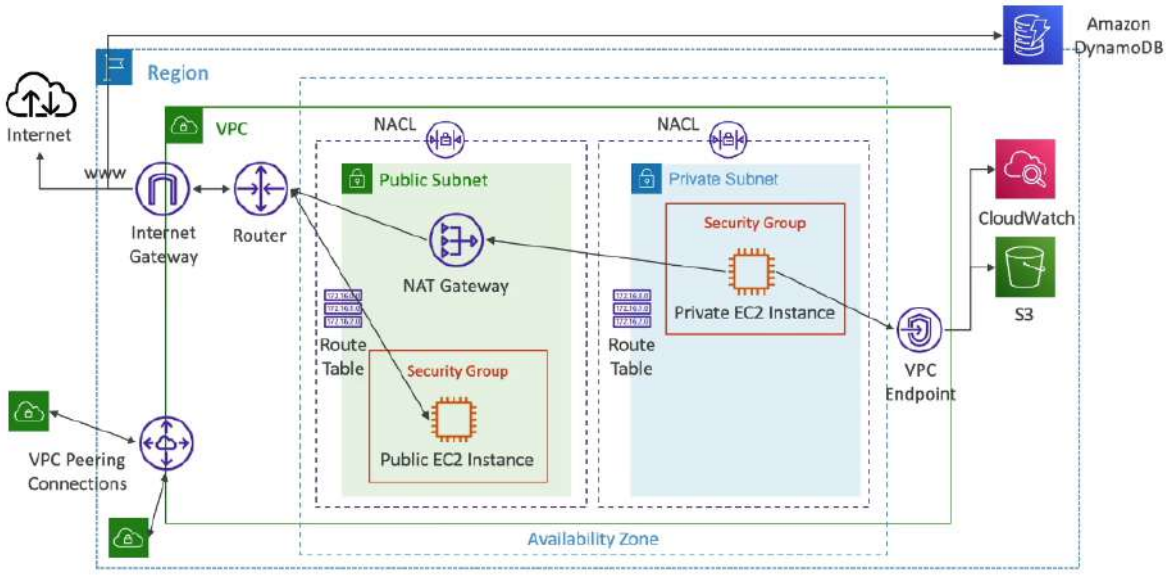


VPC Endpoints (powered by AWS PrivateLink) allow us to connect to AWS Services using a private network instead of the public internet. It gives us enhanced security and lower latency to access AWS services.

- **Interface Endpoints** (powered by PrivateLink) - provisions an ENI (private IP address) as an entry point (must attach a Security Group). It supports most AWS services. We pay per hour and per GB of data processed.
- **Gateway Endpoints** - provisions a gateway and must be used as a target in a route table (does not use security groups). It supports S3 and DynamoDB and it is free.

  We can attach an endpoint policy that controls access to the service to which we are connecting. We can modify the endpoint policy attached to our endpoint and add or remove the route tables used by the endpoint. An endpoint policy does not override or replace IAM user policies or service-specific policies.





▼ ☑ **VPC Flow Logs.**

VPC Flow Logs capture information about IP traffic going into our interfaces.
There are also Subnet Flow Logs and ENI Flow Logs. VPC Flow logs data can go to S3, CloudWatch Logs and Kinesis Data Firehose.

srcaddr & dstaddr - identify problematic IP.

srcport & dstport - identify problematic ports.

action - success or failure of the request due to Security Group/NACL.

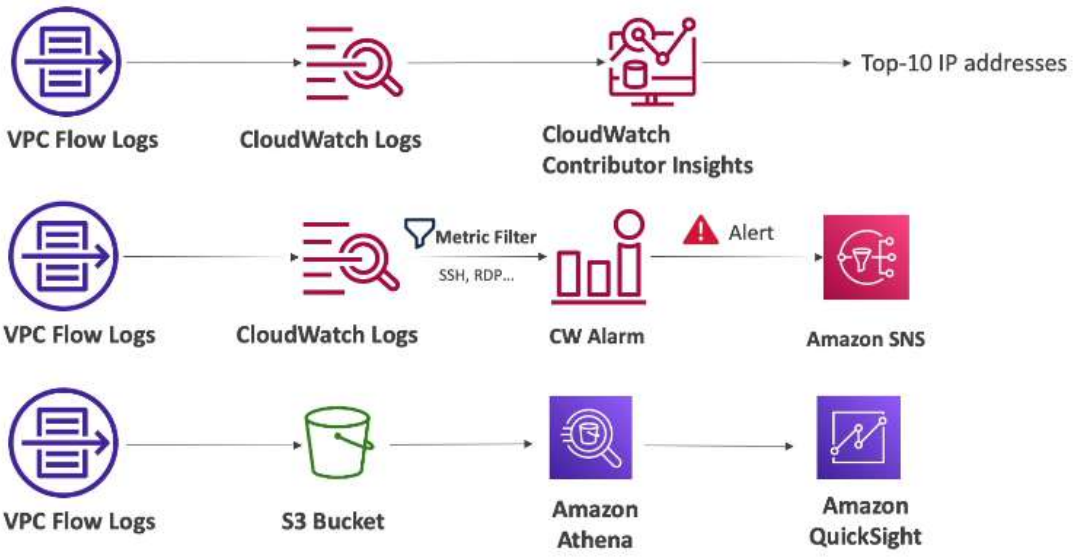
We can query VPC Flow Logs using Athena on S3 or CloudWatch Logs Insights.

Troubleshooting SG & NACL

We can look at the "ACTION" field to troubleshoot problems.

- Inbound REJECT ⇒ NACL or SG problem.
- Inbound ACCEPT, Outbount REJECT ⇒ NACL problem.
- Outbound REJECT ⇒ NACL or SG problem.
- Outbount ACCEPT, Inbound REJECT ⇒ NACL problem.

▼ **VPC Flow Logs Architectures.**



▼ ☐ **Site-to-Site VPN, VGW & CGW.**

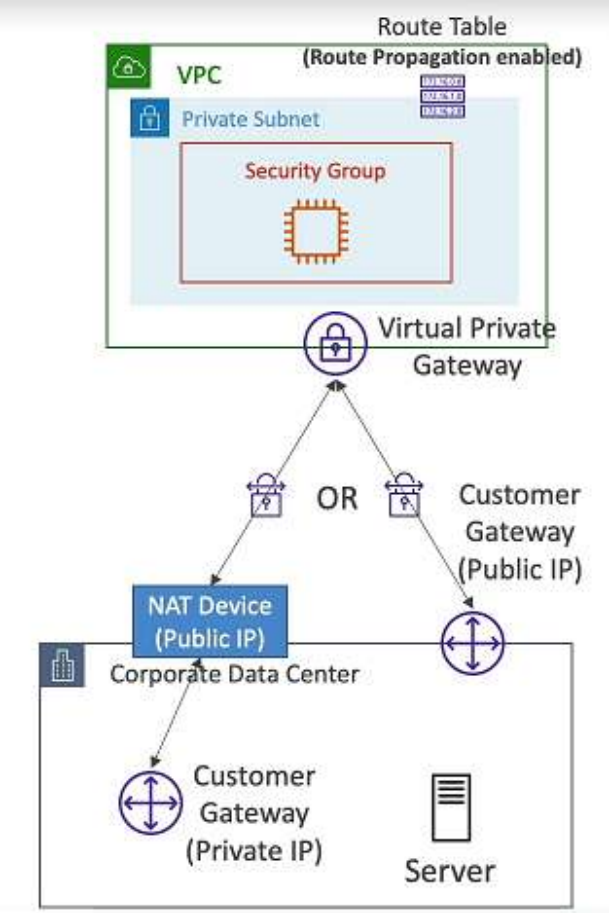
Site-to-Site VPN is used to connect an on-premises VPN to AWS. The connection is automatically encrypted and it goes over the public internet.

To establish Site-to-Site VPN:

- **On-premises** must use a **Customer Gateway (CGW)** - a physical device or software application on the customer's side of the VPN connection. It can be a hardware VPN device, such as a router or firewall, or it can be software-based.
- **AWS** must use a **Virtual Private Gateway (VGW)** - the VPN concentrator on the Amazon side of the VPN connection within the AWS infrastructure. It is a logical entity that represents a VPN gateway capable of handling multiple VPN connections simultaneously.

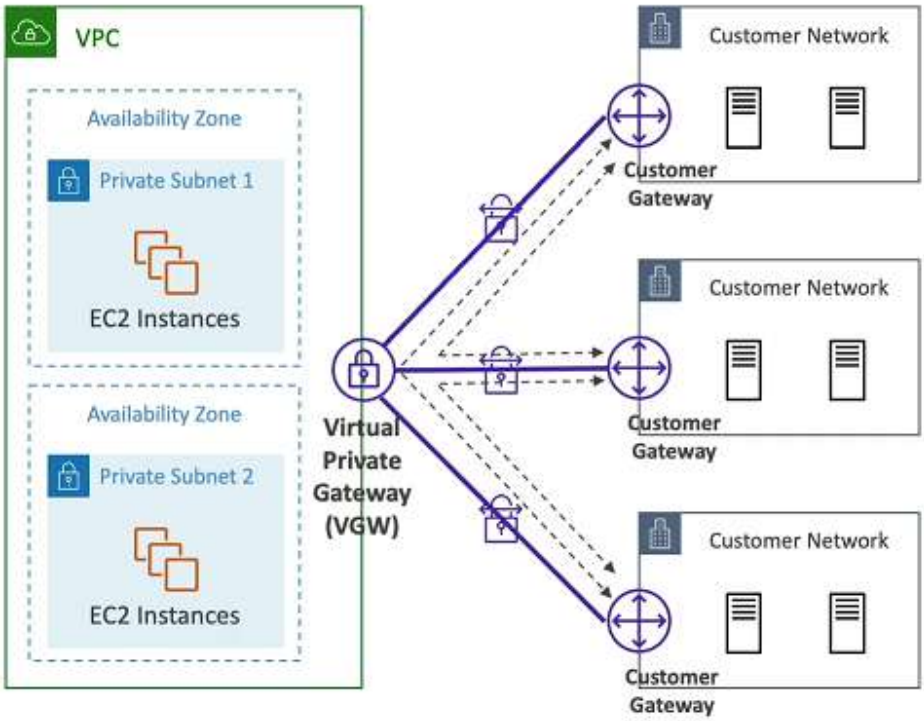
On-premises should use public internet-routable IP address for its CGW device or if it's behind a NAT device that is enabled for NAT traversal (NAT-T), it should use the public IP address of the NAT device.

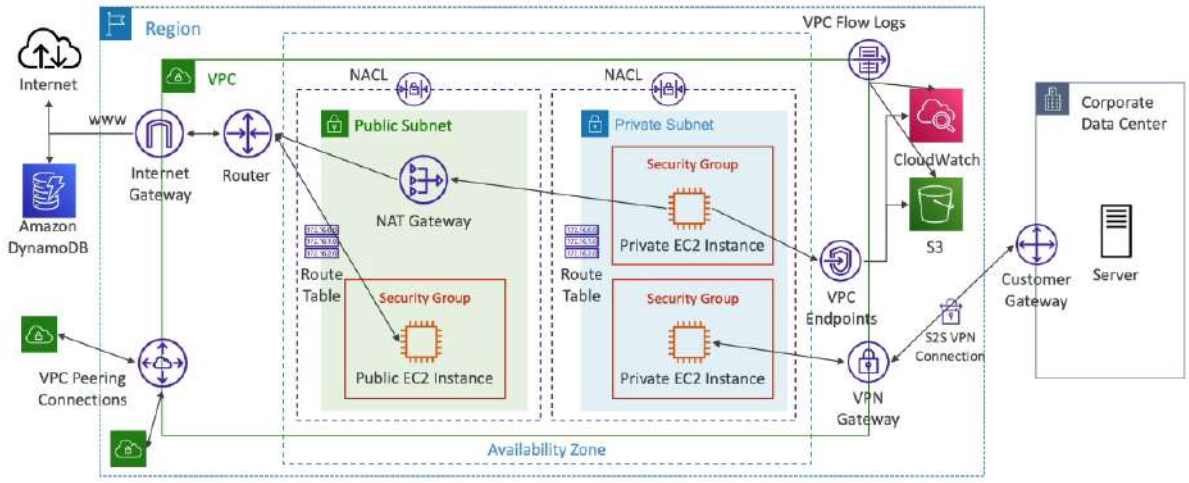
We **must enable Route Propagation for VGW in the route table that is associated with our subnets**.



> If we need to ping our EC2 instances from on-premises, we need to add the ICMP protocol on the inbound of our security groups.

AWS VPN CloudHub - provides secure communication between multiple sites, if we have multiple VPN connections. To set it up, we need to connect multiple VPN connections on the same VGW, setup dynamic routing and configure route tables.





▼ ☐ DX & DX Gateway.

Direct Connect provides a dedicated **private** connection from a remote network to our VPC. Dedicated connection must be setup between our data center and AWS DX locations. We need to setup a VGW on our VPC. With DX we can access both public and private resources on same connection.

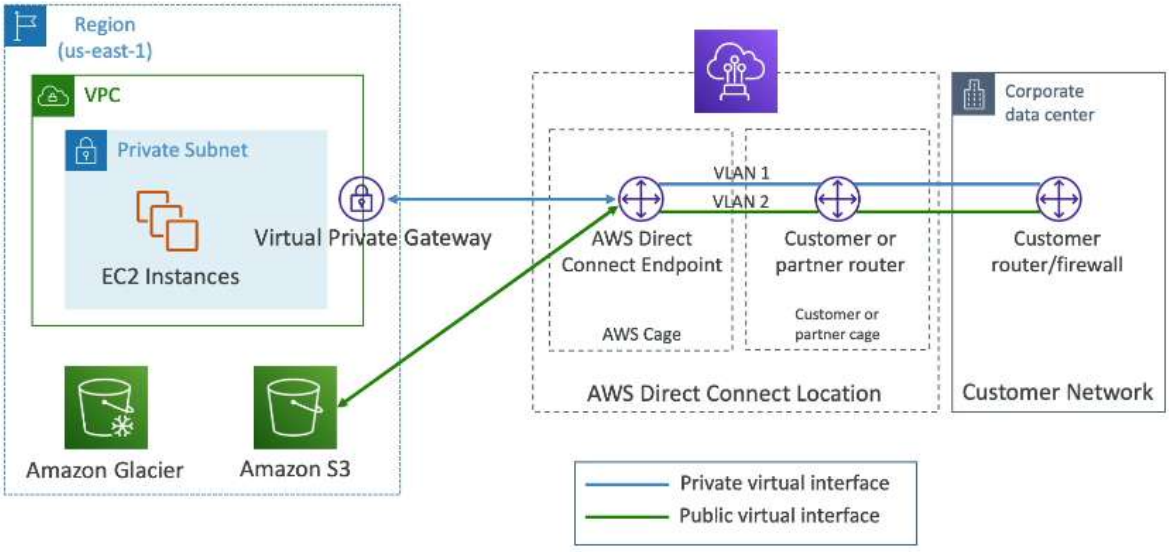
Use cases:

- Increase bandwidth throughput (**working with large data sets**).

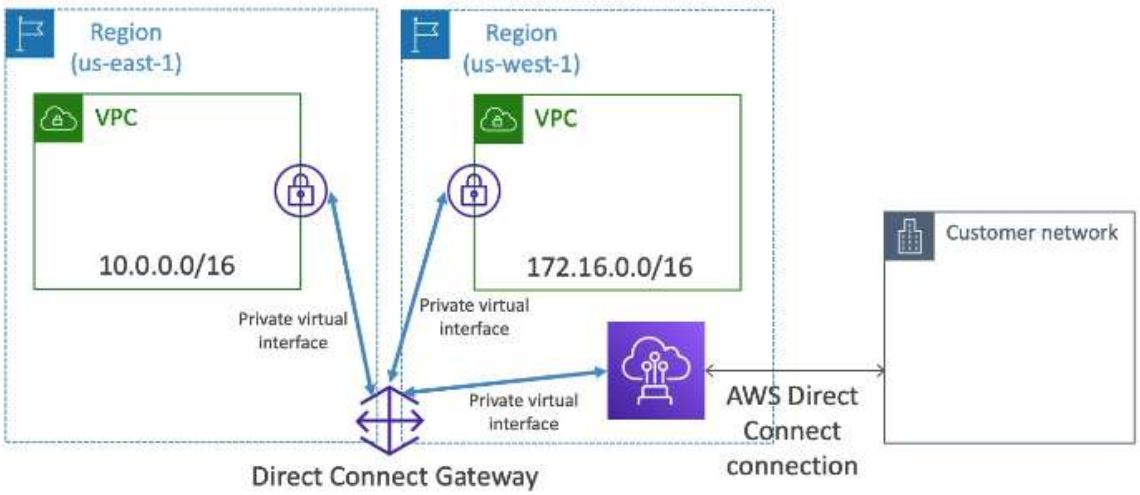
- More consistent network experience (applications using real-time data feeds).
- Hybrid environments.

Direct Connect uses **Virtual Interface** (VIFs) to logically separate different types of traffic. There are three primary types:

- **Private VIF**: Used to connect to our VPC for accessing resources like EC2 instances, RDS databases, etc. Used when we want a private link between our on-premises environment and AWS.
- **Public VIF**: Used to access AWS public services (e.g., S3, DynamoDB, etc.) over a private connection. Public VIFs connect to AWS public endpoints and can be used to route traffic to multiple AWS Regions.
- **Transit VIF**: This allows us to connect multiple VPCs across different AWS Regions using a single connection and a transit gateway.



If we want to setup a DX to one or more VPC in many different regions (same account), we must use a **DX Gateway**.



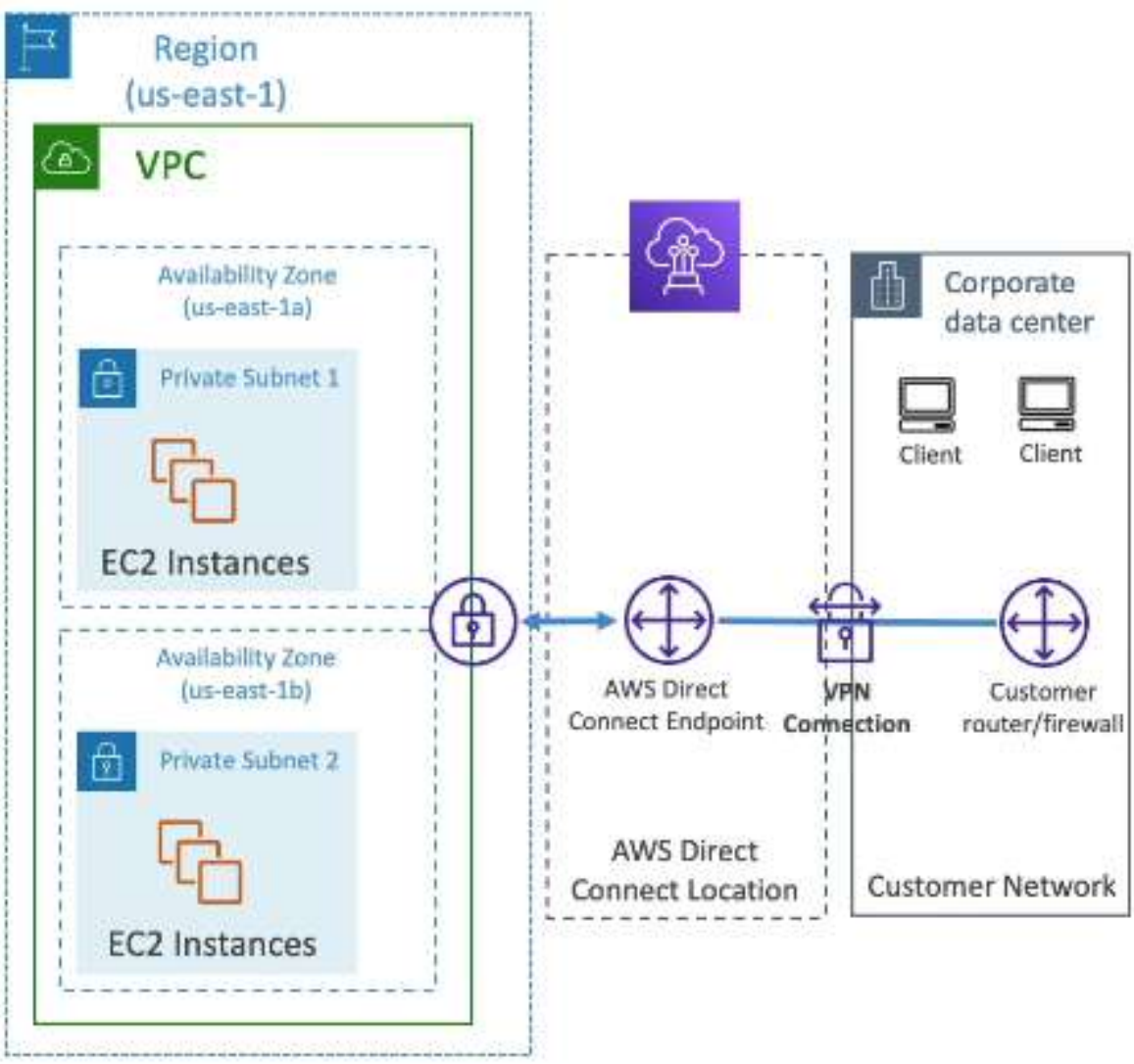
DX Connection Types

- **Dedicated Connections**: 1Gpbs, 10 Gbps, 100 Gbps capacity. A physical ethernet port is dedicated to a customer. We make request to AWS first, then it will be completed by AWS DX partners.
- **Hosted Connections**: 50Mbps, 500 Mbps, to 10 Gbps. Connection requests are made via AWS DX partners. Capacity can be added or removed on demand.

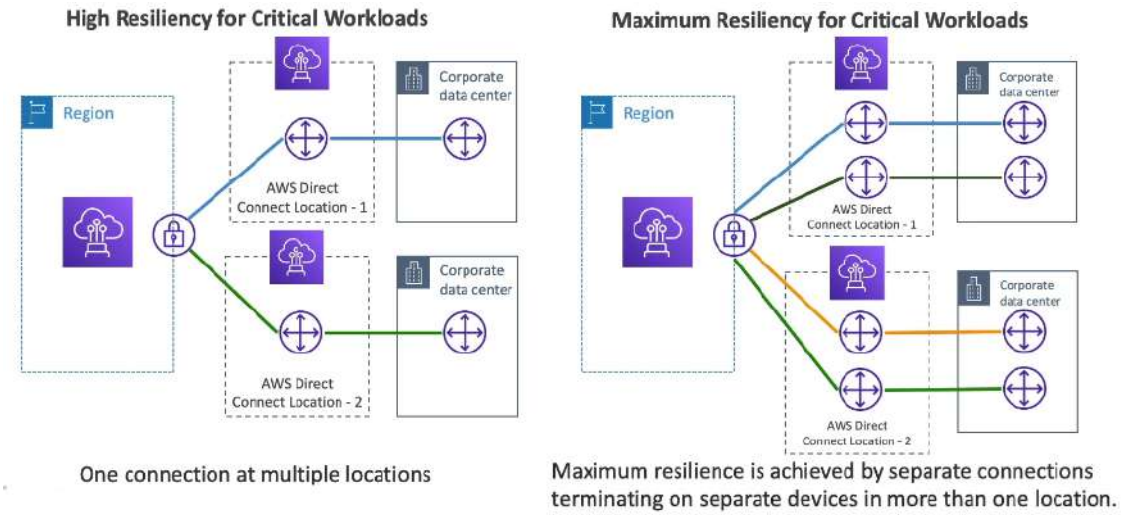
Lead times are often longer than one month to establish a new connection.

DX Encryption

Data in transit is not encrypted but is private. DX + VPN provides an IPsec-encrypted private connection.

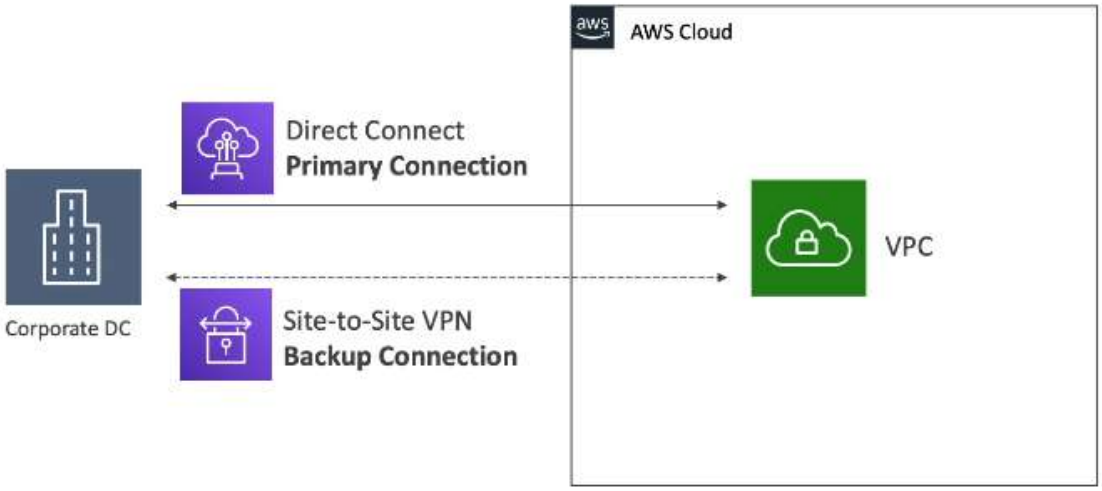


▼ **Solution Architecture - DX Resiliency Patterns.**



▼ **Solution Architecture - Site-to-Site VPN as a Backup.**

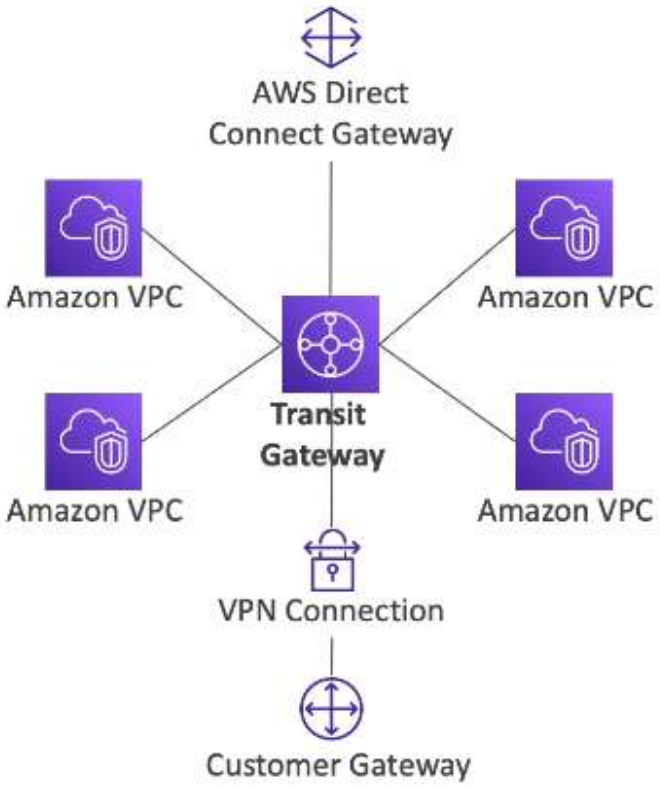
In case DX fails, we can set up a backup DX connection (expensive) or a Site-to-Site VPN connection (cheaper).



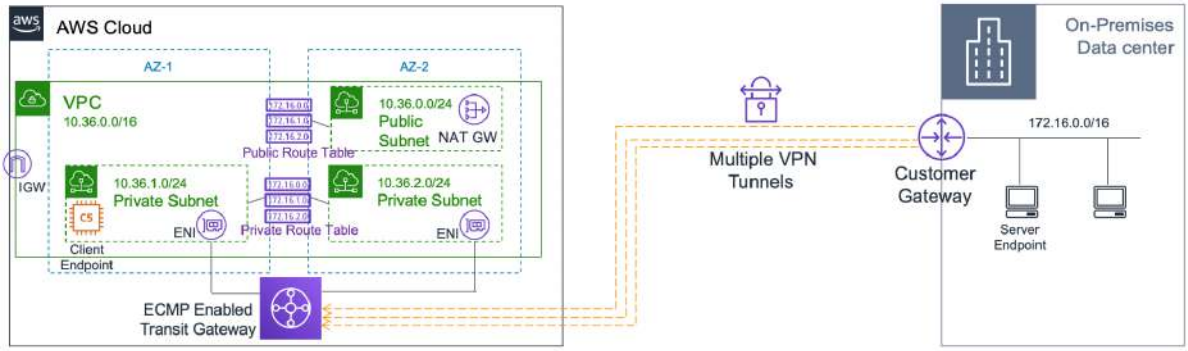
▼ ☐ **Transit Gateway.**

Transit Gateway is used for having transitive peering between thousands of VPC and on-premises. We need only one single gateway to provide this functionality. It works with DX Gateway and VPN connections. We must configure route tables to limit which VPC can talk with other VPC.

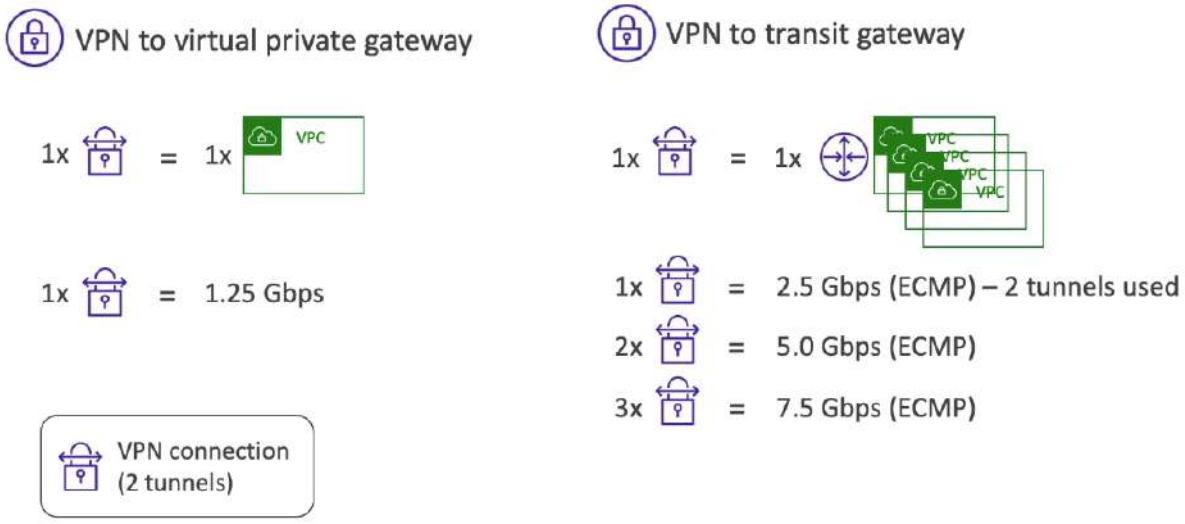
We can peer Transit Gateways across regions and share cross-account using RAM. It supports IP Multicast (**not supported by any other AWS service**).



ECMP (*Equal-cost multi-path routing*) - routing strategy that allows to forward a packet over multiple best paths. It is used to create multiple Site-to-Site VPN connections to increase the bandwidth of our connection to AWS.

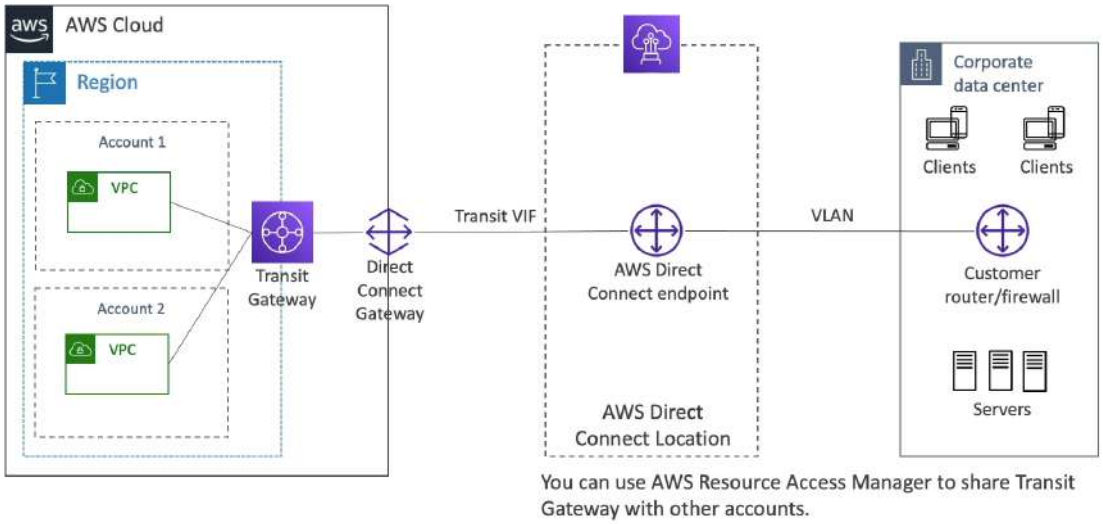


Throughput with ECMP



▼ **Solution Architecture - Share DX between Multiple Accounts.**

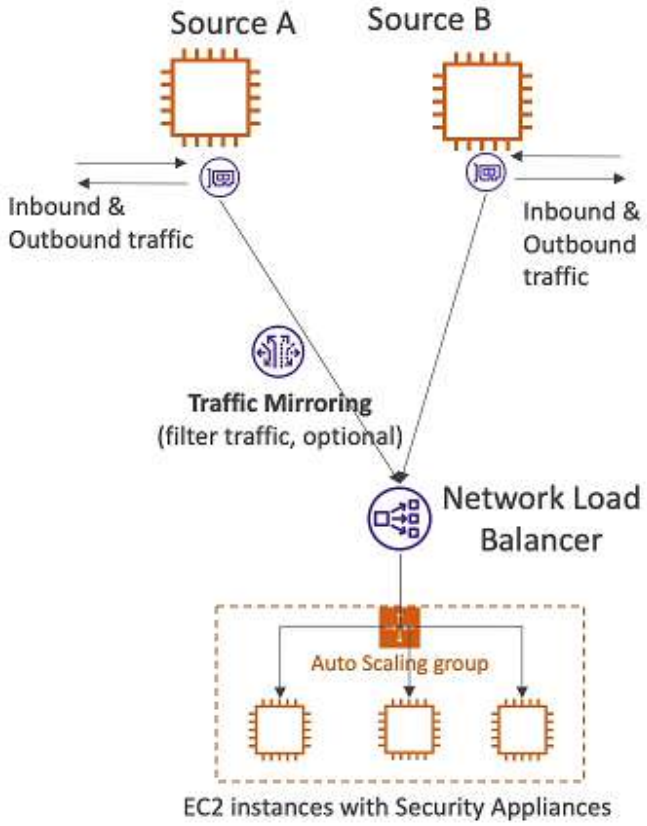
- **Cost Efficiency:** By sharing a DX connection, we reduce the cost of maintaining multiple dedicated connections.
- **Centralized Management:** It centralizes network management and simplifies configuration, especially if we have multiple AWS accounts.
- **Reduced Complexity:** It reduces the complexity of setting up and managing multiple DX connections.



▼ ☐ **VPC Traffic Mirroring.**

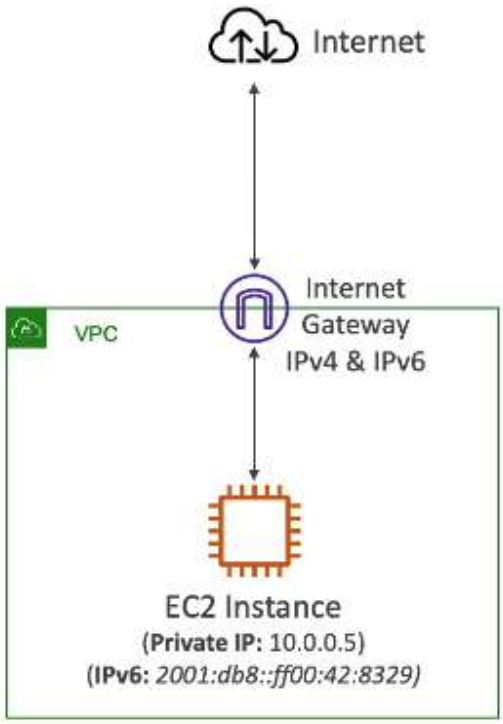
Traffic Mirroring - allows us to capture and inspect network traffic in our VPC. It routes the traffic to security appliances that we manage.

We can capture the source traffic (ENIs) and target traffic (ENI or NLB). It's possible to capture all packets or capture the packets of our interest (optionally, truncate packets). Source and target can be in the same VPC or different VPCs (VPC Peering). Traffic mirror filters can't inspect the actual packet of the incoming and outgoing traffic. It is **primarily used for monitoring and analysis**.



▼ ☑ **IPv6 for VPC.**

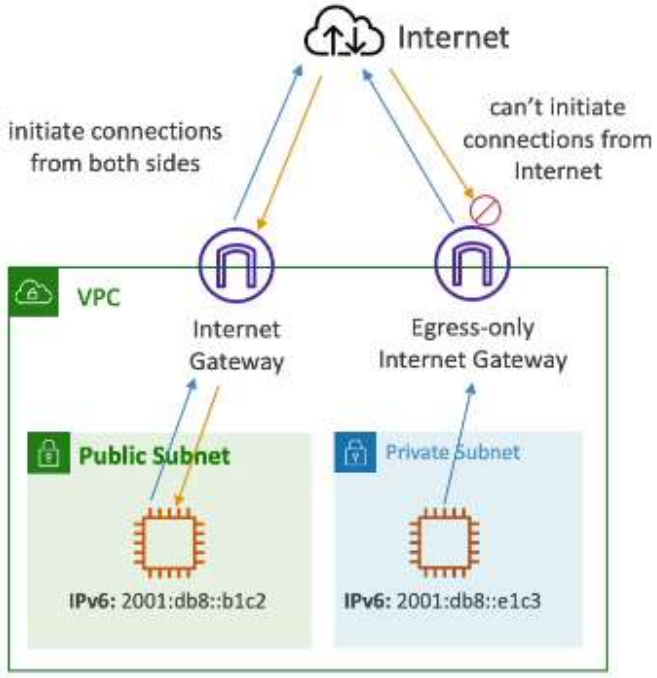
Every IPv6 address in AWS is public and Internet-routable (no private range). IPv4 cannot be disabled for our VPC and subnets. We can enable IPv6 to operate in **dual-stack mode**. Our EC2 instances will get at least a private internal IPv4 and a public IPv6.



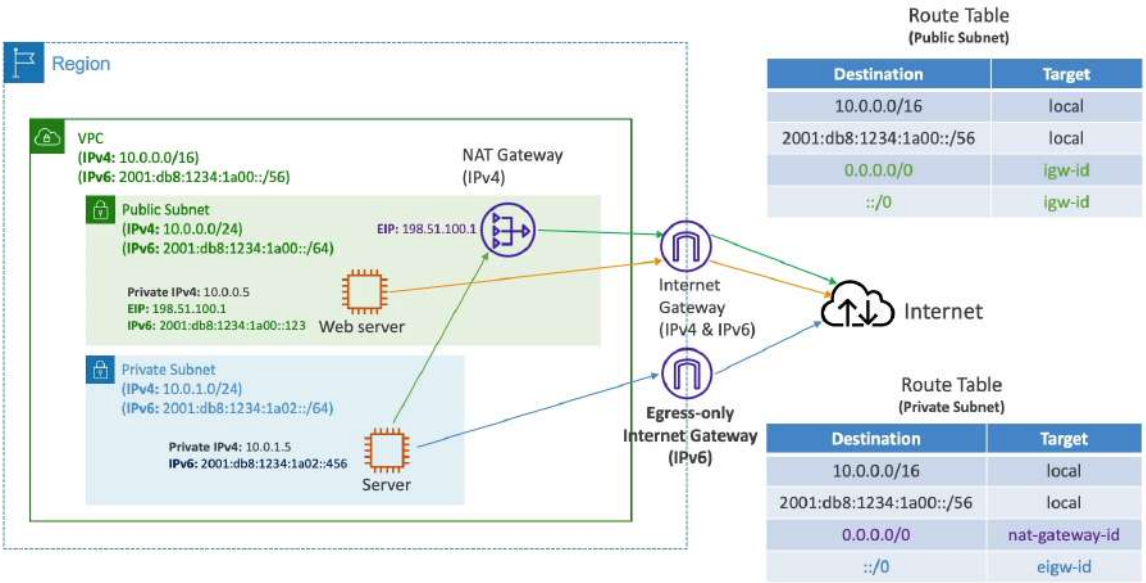
If we cannot launch an EC2 instance in our subnet. It's not because it cannot acquire an IPv6, it's because there are no available IPv4 in our subnet. To solve that we should create a new IPv4 CIDR in our subnet.

▼ ☑ **Egress-Only Internet Gateway.**

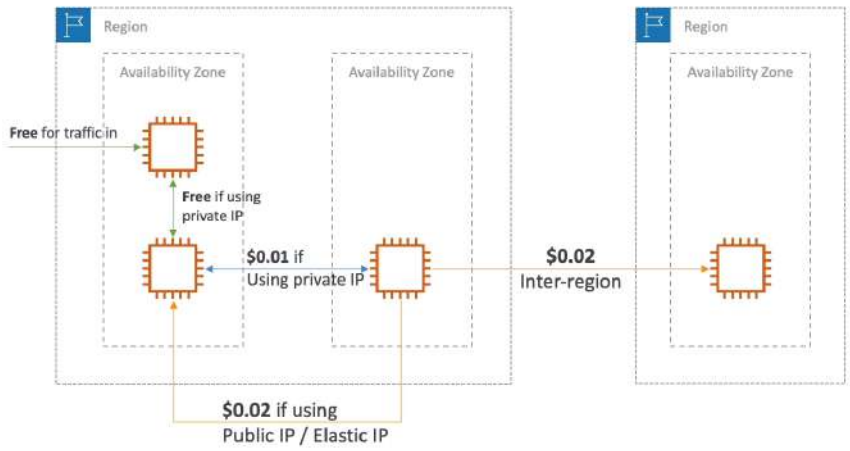
Egress-Only Internet Gateway is used for IPv6 only (similar to a NAT Gateway bur for IPv6). It allows outbound connections over IPv6 for instances in our VPC while preventing the internet to initiate an IPv6 connection to our instances. We must update the route tables.



IPv6 Routing



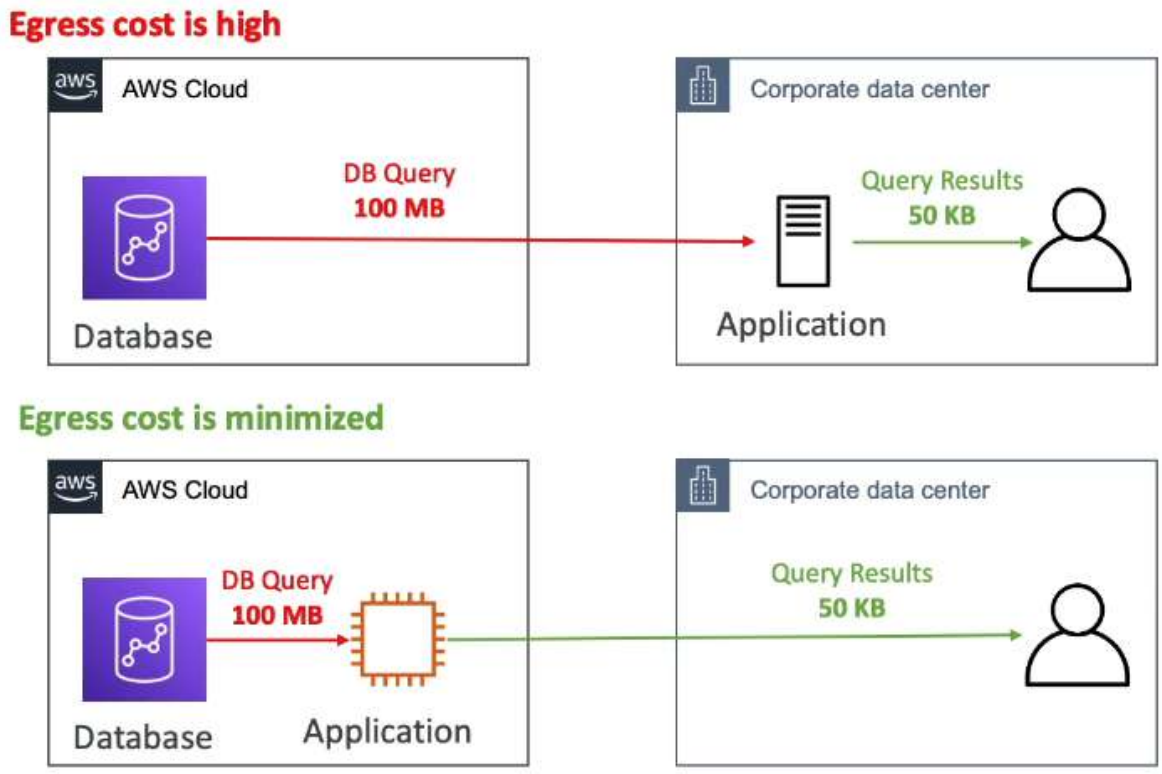
▼ ☑ **Networking Costs in AWS.**



- Use Private IP instead of Public IP for good savings and better network performance
- Use same AZ for maximum savings (at the cost of high availability)

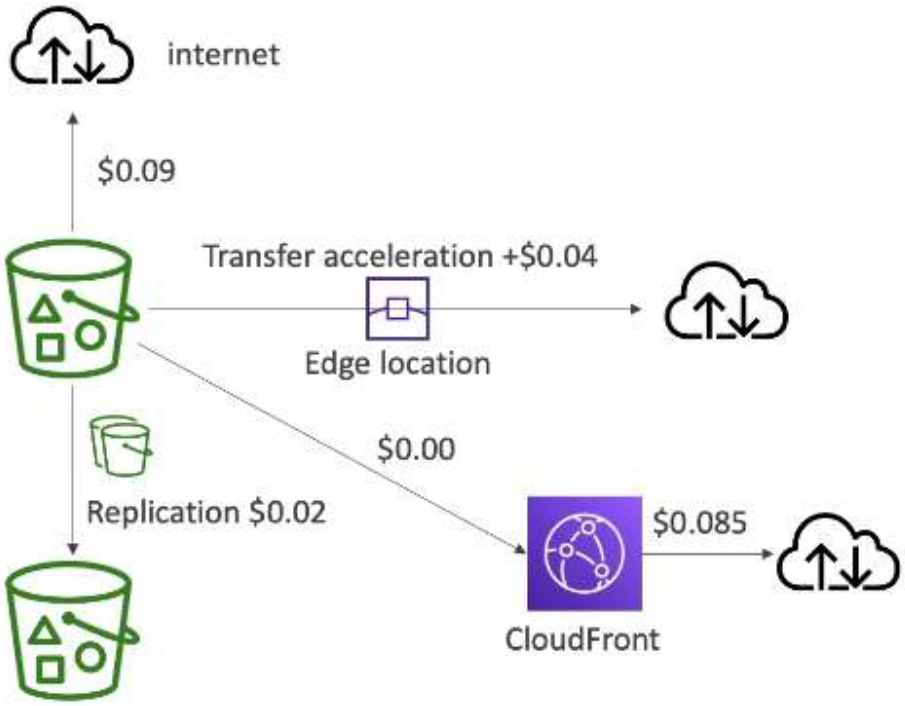
Egress traffic - outbound traffic (AWS → Outside). DX locations that are co-located in the same AWS Region result in lower cost for egress network.

Ingress traffic - inbound traffic (Outside → AWS), typically free.

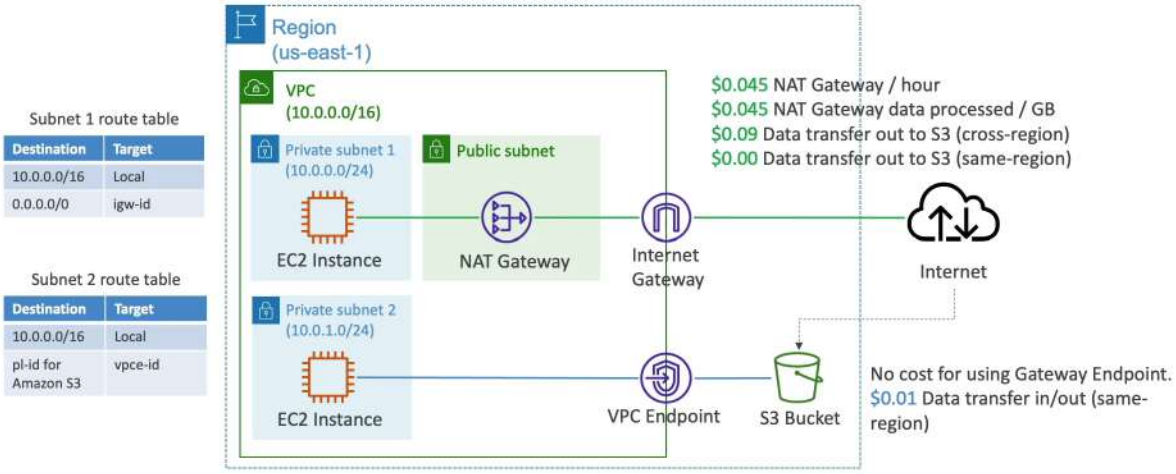


We should try to keep as much internet traffic within AWS to minimize costs.

S3 Data Transfer Pricing



NAT Gateway vs VPC Endpoint Pricing

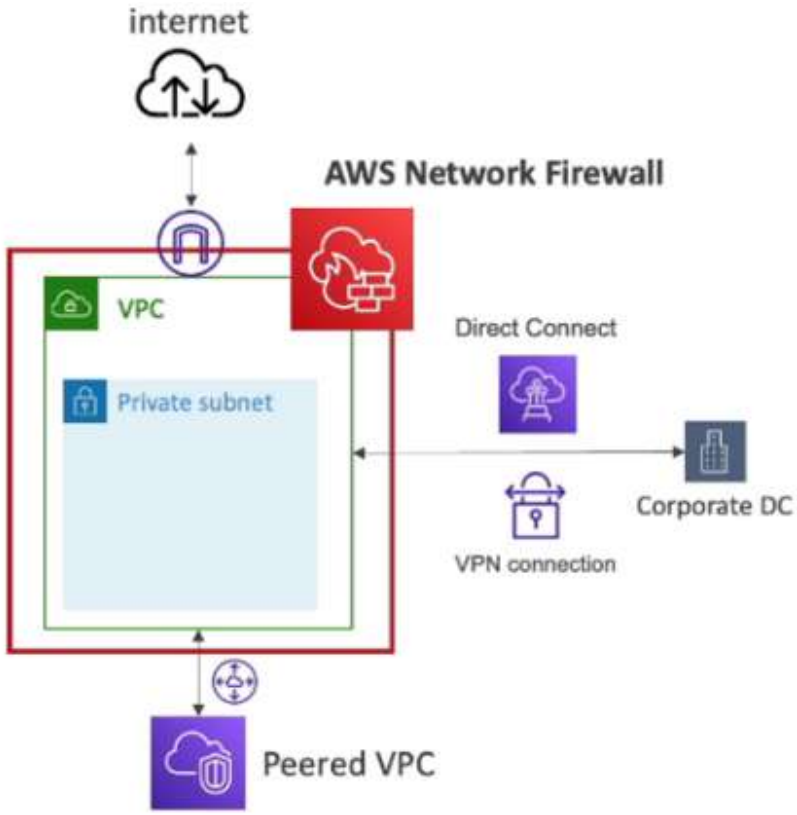


☑ AWS Network Firewall.

AWS Network Firewall protects our entire VPC (from layer 3 to layer 7). We can inspect traffic in any direction:

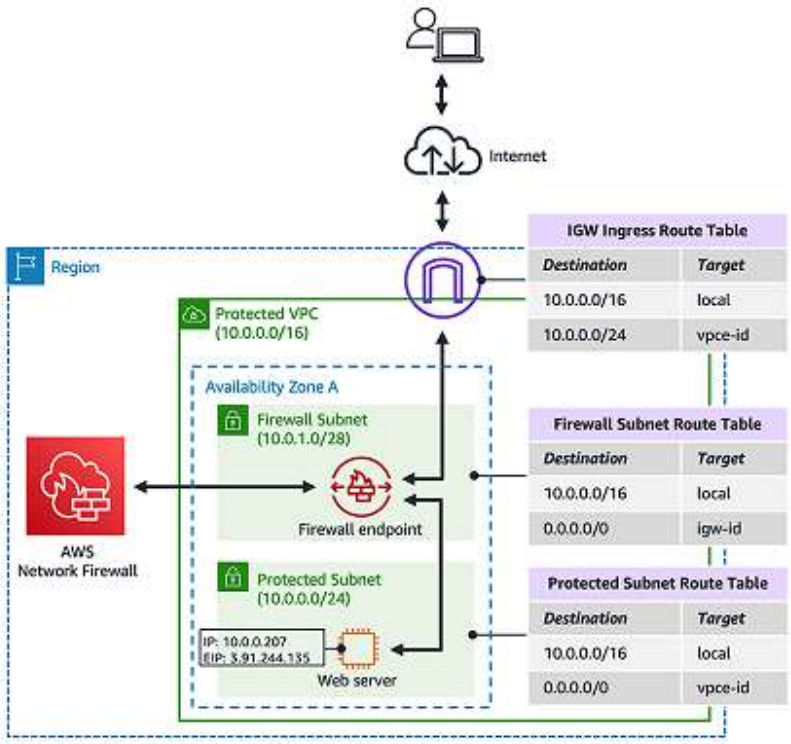
- VPC to VPC traffic
- Outbound to internet
- Inbound to internet

- To/from Direct Connect & Site-to-Site VPN



Internally, the AWS Network Firewall uses the AWS Gateway Load Balancer. Rules can be centrally managed cross-account to apply to many VPCs.

Network Firewall Endpoint - a virtual appliance deployed within our VPC that inspects and filters network traffic based on the firewall rules we configure. It provides the actual interface through which the firewall processes the traffic. It's responsible for applying the policies and rules we define to **control incoming and outgoing network traffic**. We can allow, drop or alert for the traffic that matches the rules. All rule matches can be sent to S3, CloudWatch Logs and KDF.



▼ ☑ **HPC & Networking.**

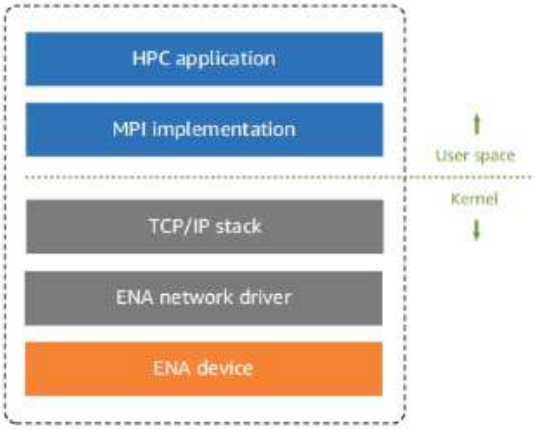
Enhanced Networking via SR-IOV (Single Root I/O Virtualization) - a technology that allows network traffic to bypass the hypervisor and directly access the network interface card (NIC).

It's typically used in instances that need high throughput and low latency but do not require the advanced features of ENA or EFA.

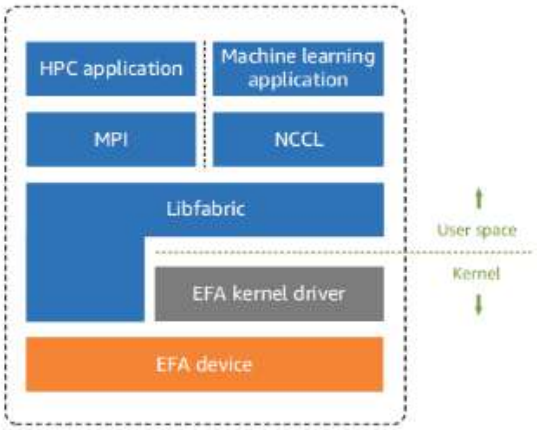
Elastic Network Adapter (ENA) - a custom network interface designed by AWS that supports high-performance networking. It offers better flexibility and advanced features compared to SR-IOV. ENA is suited for workloads that need higher network bandwidth (**up to 100 Gbps**).

Elastic Fabric Adapter (EFA) - a network interface designed for tightly coupled HPC and ML workloads that require inter-instance communication with low latency and high throughput. **EFA builds on ENA**, offering additional features like hardware-based message passing.

The OS-bypass capabilities of EFAs are not supported on Windows instances. If we attach an EFA to a Windows instance, the instance functions as an ENA without the added EFA capabilities.



Traditional HPC software stack in EC2



HPC software stack in EC2 with EFA

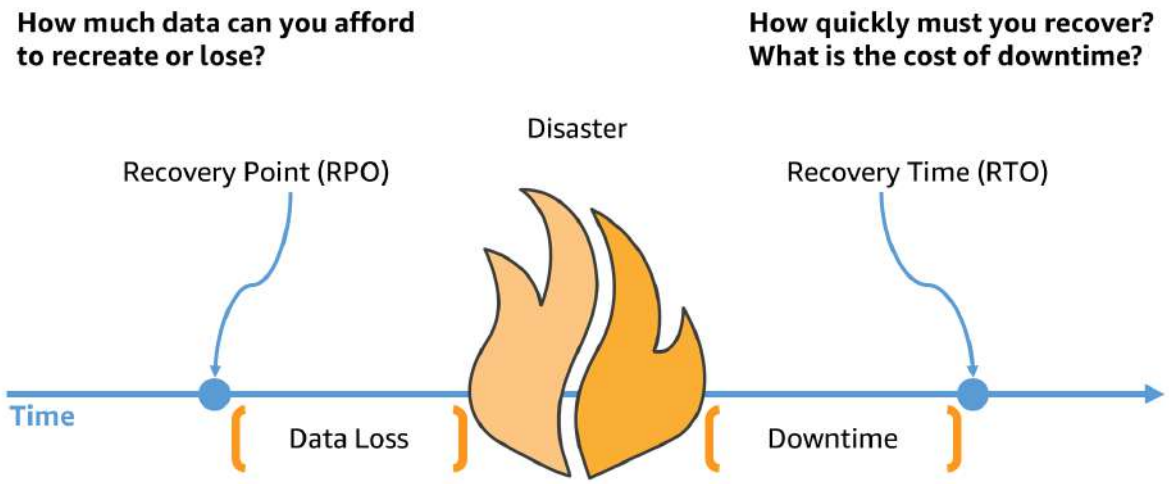
AWS ParallelCluster is an open-source cluster management tool that simplifies the deployment and management of HPC clusters in the AWS Cloud. It allows us to easily set up a multi-node cluster for our HPC workloads. There is an ability to enable EFA on the cluster and improve network performance.

Disaster Recovery & Migrations

▼ ☐ **Disaster Recovery in AWS.**

RPO (Recovery Point Objective) - indicates the maximum amount of data that can be lost measured in time. It's the point in time to which your data must be recovered after an incident.

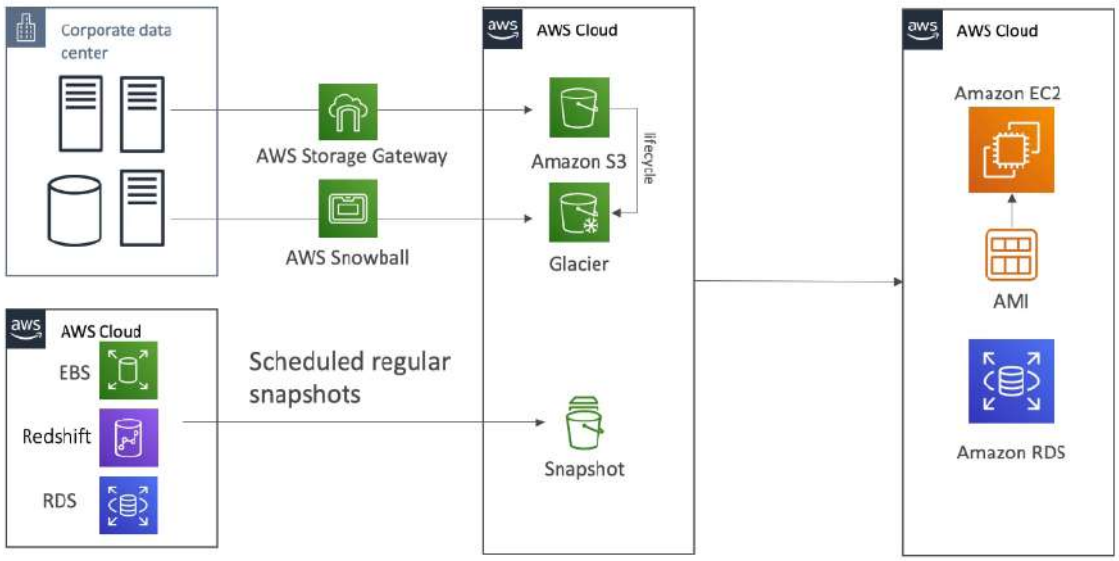
RTO (Recovery Time Objective) - maximum acceptable time that it takes to restore a system or service after a disruption. It reflects how quickly you need to recover after a failure.



Disaster Recovery Strategies

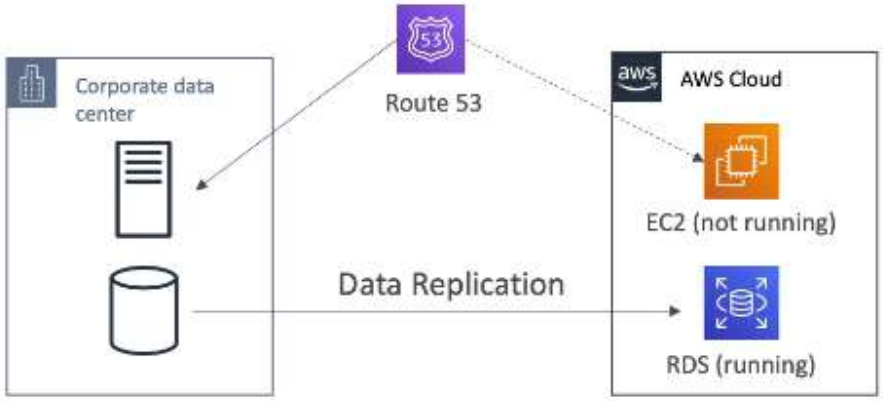
- **Backup & Restore** - most basic DR strategy where data is backed up to AWS storage services and restored in the event of a disaster. Suitable for non-critical applications where downtime can be tolerated.

  RTO and RPO are typically longer because the restoration process involves re-provisioning infrastructure and retrieving data.

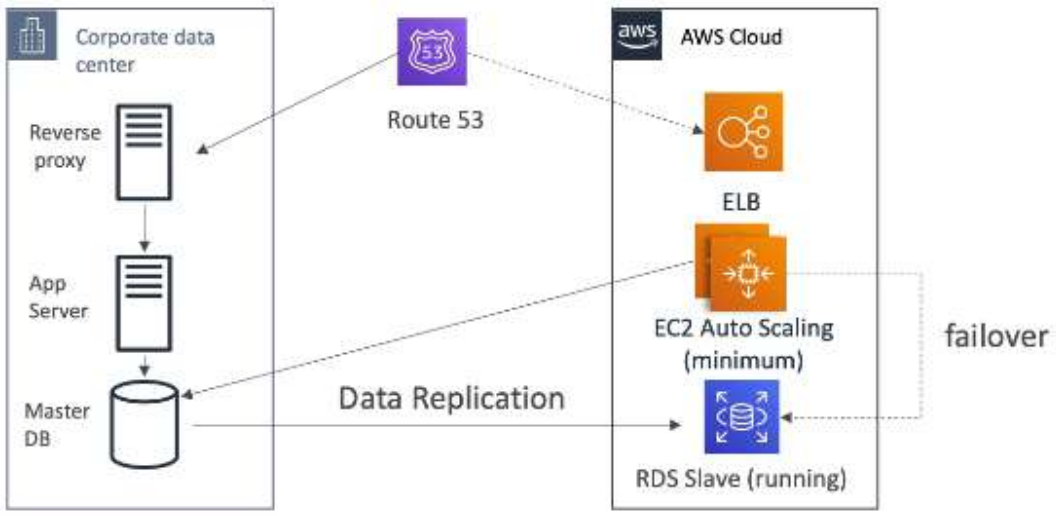


- **Pilot Light** - a minimal version of an environment is always running in AWS, which can be scaled up to a full-scale production environment when needed. Suitable for critical applications where a minimal version can handle some traffic until full scaling occurs.

Faster than the backup and restore strategy but still involves some time for scaling up and provisioning additional resources. The RPO can be near real-time since core components are active.

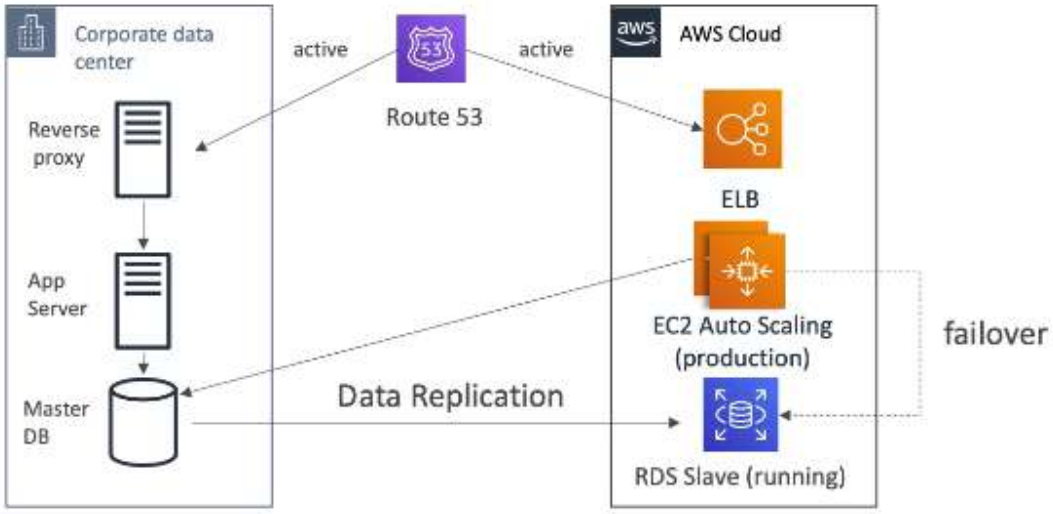


- **Warm Standby** - a scaled-down version of a fully functional environment is running in AWS, which can be quickly scaled up to handle production load. Suitable for critical applications requiring reduced RTO and RPO.



- **Multi-Site (Hot-Site)** - the most robust DR strategy where multiple active sites run simultaneously, providing full redundancy and load balancing. Suitable for mission-critical applications requiring near-zero downtime and high availability.

  This approach offers the lowest RTO and RPO, potentially near zero, as both sites are live and in sync.



Disaster Recovery Strategies Comparison

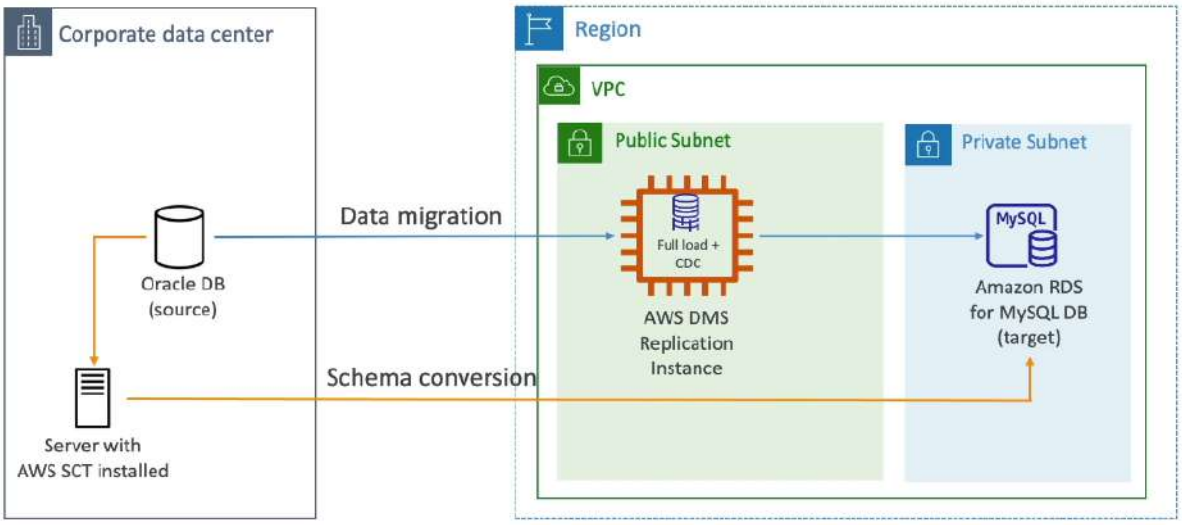
Strategy	RTO	RPO	Cost	Complexity	Typical Use Case
Backup and Restore	Hours	Hours	Low	Low	Low-criticality workloads with tolerant downtime
Pilot Light	Minutes	Minutes	Moderate	Moderate	Critical apps needing quicker recovery but cost-sensitive
Warm Standby	Minutes	Near real-time	Higher	Moderate	Important applications requiring minimal downtime
Multi-Site	Seconds	Near zero	High	High	Mission-critical apps demanding high availability

▼ ☑ **DMS.**

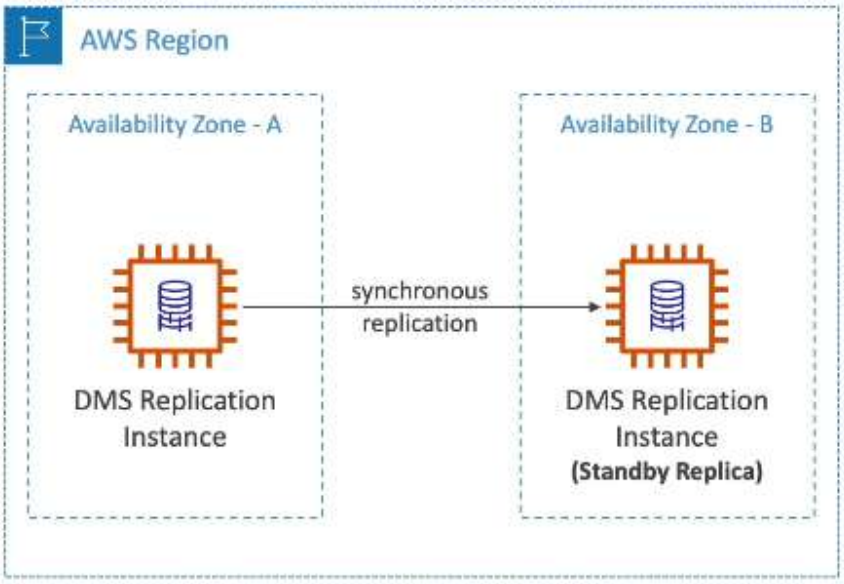
With **DMS** we can quickly and securely migrate databases to AWS. The source database remains available during the migration. DMS supports both homogeneous and heterogeneous migrations. It can migrate data from various sources like on-premises databases or other AWS services. DMS includes a schema conversion tool (AWS SCT) that helps convert the schema of source databases to match the target database schema.

- **Homogeneous migrations** - involves moving data between two systems that have the same DBMS.
- **Heterogeneous migrations** - involves moving data between two systems that have different DBMS.

It can be used for continuous data replication with minimal latency (keep source and target databases in sync for ongoing data changes).

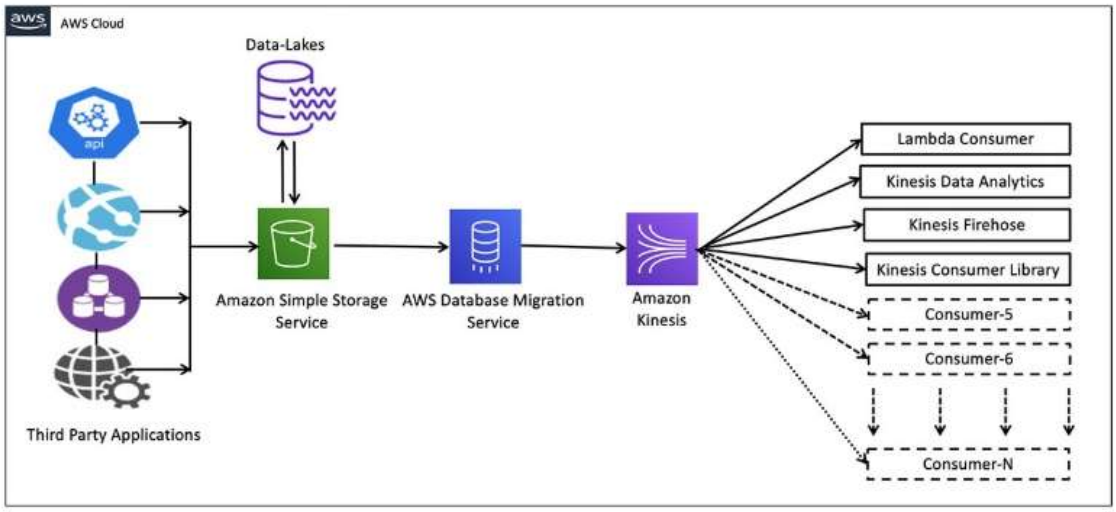


When Multi-AZ is enabled, DMS provision and maintains a synchronously stand replica in a different AZ. It eliminates I/O freezes and provides data redudancy.



▼ Solution Architecture - S3 to KDS

AWS DMS lets us expand the existing application to stream data from Amazon S3 into Amazon Kinesis Data Streams for real-time analytics without writing and maintaining new code.



▼ ☑ **RDS & Aurora Migrations.**

RDS MySQL → Aurora MySQL

- **Option 1**: DB Snapshots from RDS MySQL restored as MySQL Aurora DB.
- **Option 2**: Create an Aurora Read Replica from our RDS MySQL and when the replication lag is 0, promote it as its own DB cluster (can take time and money).

External MySQL → Aurora MySQL

- **Option 1**: Use Percona XtraBackup to create a file backup in S3 and then create an Aurora MySQL DB from S3.
- **Option 2**: Create an Aurora MySQL DB and use mysqldump utiliy to migrate MySQL into Aurora (slower than option 1).

RDS Postrgres → Aurora Postgres

- **Option 1**: DB Snapshots from RDS Postgres restored as Postgres Aurora DB.
- **Option 2**: Create an Aurora Read Replica from our RDS Postgres and when the replication lag is 0, promote it as its own DB cluster (can take time and money).

External Postgres → Aurora Postgres

- Create backup and put it in S3 then import it using the aws_s3 Aurora extension.

If both databases are up and running, use DMS.

▼ ☑ On-premises Strategies with AWS.

- **Amazon Linux 2 AMI as VM** (.iso format)
- **VM Import/Export** - migrate existing applications into EC2, create a DR repository strategy for our on-premise VMs. Can export back the VMs from EC2 to on-premise.
- **AWS Application Discovery Service** - gather information about our on-premise servers to plan a migration.
- **AWS Database Migration Service** - replicate on-premise ⇒ AWS, AWS ⇒ AWS, AWS ⇒ on-premise.
- **AWS Server Migration Service** - incremental replication of on-premise live servers to AWS.

▼ ☑ AWS Backup.

AWS Backup is a fully-managed service to centrally manage and automate backups across AWS services. It can have on-demand and scheduled backups. Supports PITR.

Cross-Region Backup - backing up data from one AWS region to another. This is useful for disaster recovery scenarios where we want to ensure data availability even if an entire AWS region becomes unavailable due to a disaster or outage.

Cross-Account Backup - backing up data from AWS resources in one AWS account to AWS Backup vaults in another AWS account. This is useful for scenarios where we want to centralize backups for multiple AWS accounts in a single backup location.

Backup Vault Lock - enforces WORM state for all the backups that we store in our AWS Backup Vault. It is additional layer of defense to protect our backups against inadvertend or malicious delete operations. Even root user cannot delete backups when enabled.

▼ ☑ Application Discovery Service & Application Migration Service.

Application Discovery Service helps enterprises in planning their migration to AWS by discovering on-premises applications and **collecting configuration, usage, and behavior data**. It provides insights into application dependencies and resource utilization.

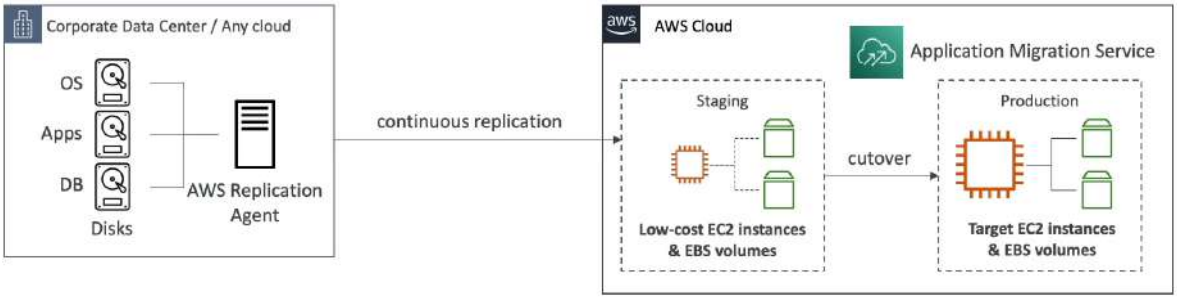
Agentless Discovery (**AWS Agentless Discovery Connector**) - uses network-based data collection methods to gather information about on-premises servers and applications without requiring the installation of agents.

Agent-based Discovery (**AWS Application Discovery Agent**) - uses lightweight agents installed on on-premises servers to collect detailed data about system configurations, running processes, network connections, and performance metrics.

Resulting data can be viewed within AWS Migration Hub.

Application Migration Service (MGN) simplifies and the migration of applications from physical, virtual, and cloud infrastructure to AWS. It automates the migration process, reducing the time, cost, and effort involved in traditional lift-and-shift migrations.

Implementation begins by installing the AWS Replication Agent on our source servers. When we launch Test or Cutover instances, AWS Application Migration Service automatically converts our source servers to boot and runs natively on AWS.



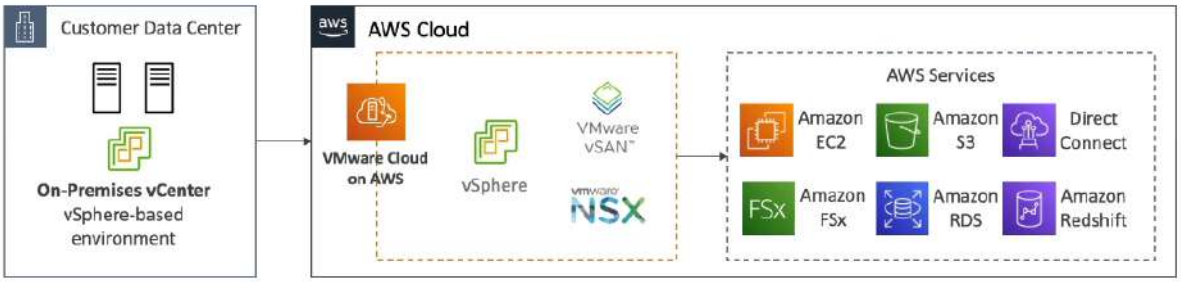
▼ ☑ **Transfering Large Datasets into AWS.**

- **Over the internet/Site-to-Site VPN** - ideal for small to moderate data volumes where latency isn't a significant concern.
- **Over Direct Connect** - suitable for transferring large datasets where consistent, high-throughput, and low-latency connectivity is required.
- **Over Snowball** - ideal when dealing with extremely large datasets or when we have limited or unreliable internet connectivity.
- **For on-going replication/transfers** - Site-to-Site VPN/DX with DMS or DataSync.

▼ ☑ VMware Cloud on AWS.

Some customers use VMware Cloud to manage their on-premises Data Center. They want to extend Data Center capacity to AWS, but keep using the VMware Cloud software.

VMware Cloud on AWS is a service that allows businesses to run VMware's virtualization technologies on AWS infrastructure. Essentially, it extends our VMware environment to the AWS cloud

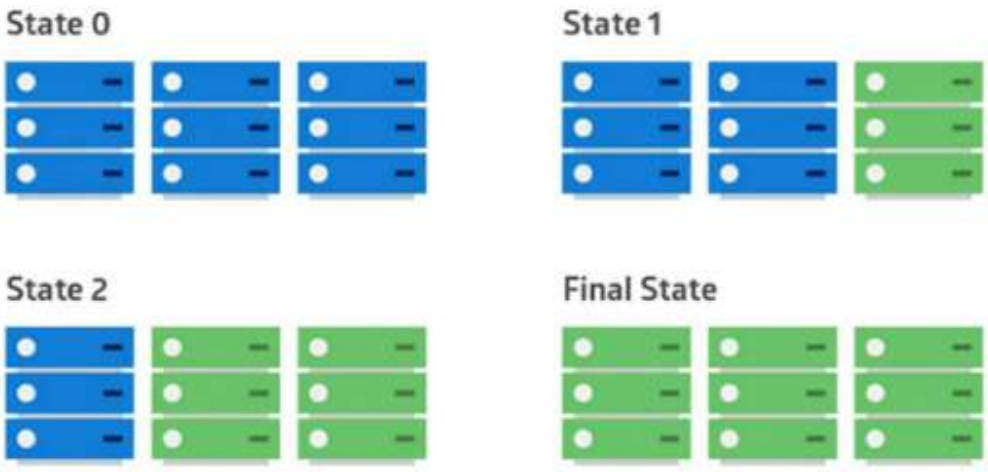


▼ ☐ AWS Deployment Strategies.

All-at-Once (Big Bang) - deploys the new version of the application to all instances at once. It is simple and fast, with no need for complex infrastructure setups. It has high risk of downtime if the new version has issues, no easy rollback. If something goes wrong, the entire system can be affected.

Use Cases: **Non-critical applications where downtime is acceptable or where rollback procedures are straightforward**.

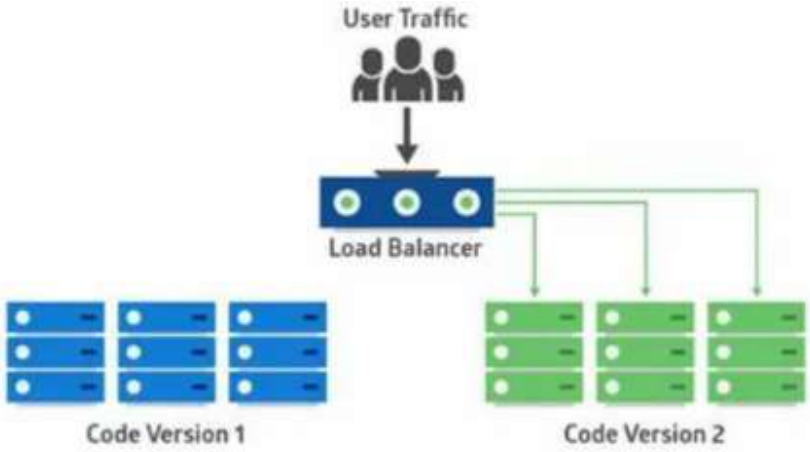
Rolling Deployment - gradually replaces instances of the old version with instances of the new version. Reduces the risk of full-scale failure and can be quickly fixed or rolled back.



Use Cases: **Applications that need to minimize downtime but can tolerate temporary inconsistencies**.

Blue-Green Deployment - two identical environments are maintained: one (blue) running the current version and another (green) running the new version. Traffic is switched from the blue environment to the green one once the new version is ready.

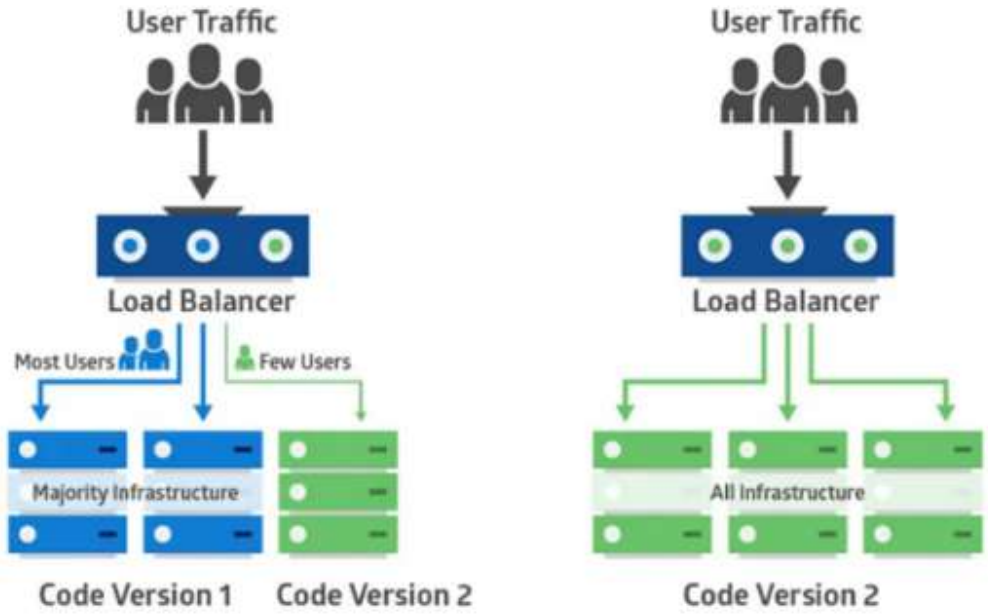
It is zero-downtime deployment, easy rollback by switching traffic back to the blue environment. Testing and validation can be done on the green environment.



Use Cases: **Applications requiring zero downtime and where rollback needs to be swift and reliable**.

Canary Deployment - small portion of the traffic is directed to the new version while the rest remains on the old version. Gradually, more traffic is shifted to the

new version as confidence in its stability grows. It allows for real-world testing with minimal risk.



Use Cases: Applications with a **large user base or critical services where early feedback and incremental rollout are important**.

A/B Testing Deployment - similar to Canary deployment, but with a focus on running two or more versions simultaneously to compare performance or user experience. Different users are directed to different versions.

Use Cases: **Testing new features, UI/UX changes, or optimizations**.

Shadow Deployment - new version is deployed alongside the old version, but the new version does not serve production traffic. Instead, it processes a copy of the production traffic to see how it behaves.

Use Cases: Validating major updates or entirely new services without risking production stability.

Other Services

▼ ☑ **CloudFormation.**

CloudFormation is a declarative way of outlining our AWS Infrastructure, for any resources.

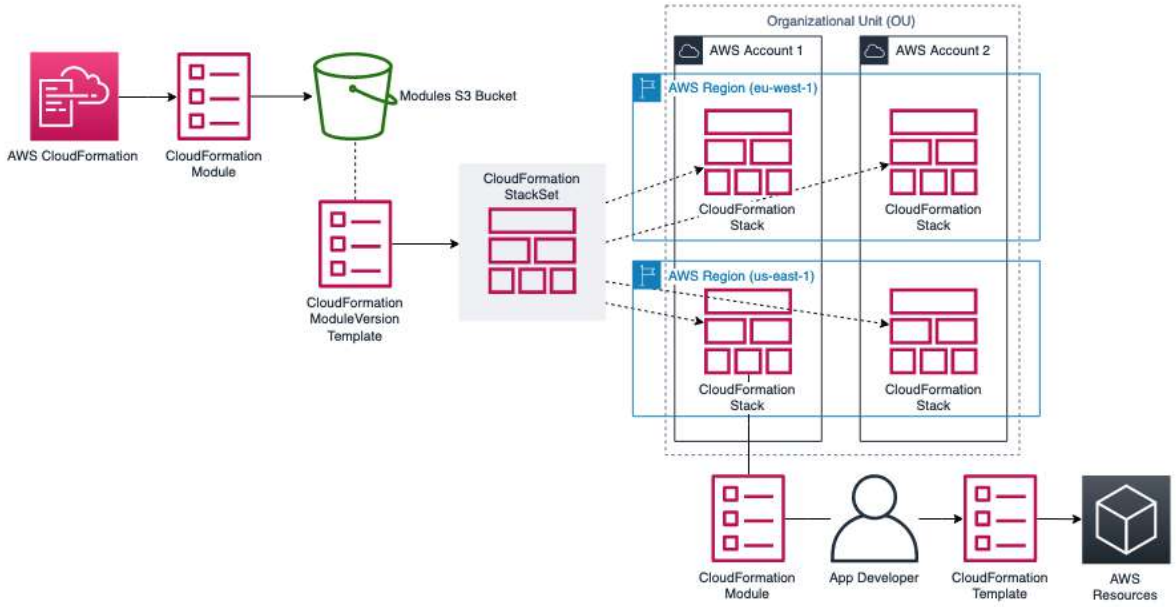
> Example: We want security group, two EC2 instances using that security group, an S3 bucket and ELB in front of these machines. CloudFormation creates all those for us, in the right oreder, with exact configuration that we specify.

Benefits of CloudFormation

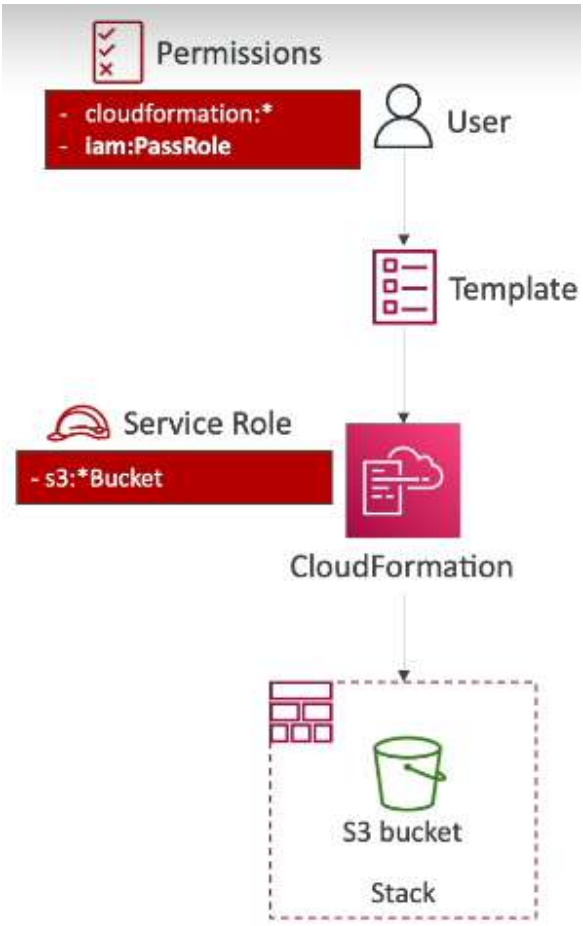
- Infrastructure as code - No resources are manually created, which is excellent for control. Changes to the infrastructure are reviewed through code.
- Cost - Each resources within stack is tagged with an identifier so we can easily see how much a stack costs us. We can estimate the costs of our resources using the CloudFormation template.
- Productivity - Ability to destroy and re-create an infrastructure on the cloud on the fly. We can automate generation of diagrams for our templates.

Application Composer - A visual tool designed to simplify the process of building serverless applications on AWS. It enables developers to drag and drop various AWS services to create an architecture diagram that represents their application.

- **Stacks** are the core unit of deployment in CloudFormation. A stack is a collection of AWS resources that we can manage as a single unit. When you create, update, or delete a stack, CloudFormation automatically provisions or deletes the associated resources in a predictable way.
- **Template** is a JSON or YAML formatted text file that describes the resources AWS CloudFormation will provision in our stack.
- **StackSets** extend the functionality of stacks by allowing us to manage stacks across multiple accounts and regions.



CloudFormation Service Role - IAM role that allows CloudFormation to create/update/delete stack resources on our behalf. It is useful if we want to achieve the least privilege principle, but we dont want to give the user all the required permissions to create the stack resource. User must have iam:PassRole permissions.

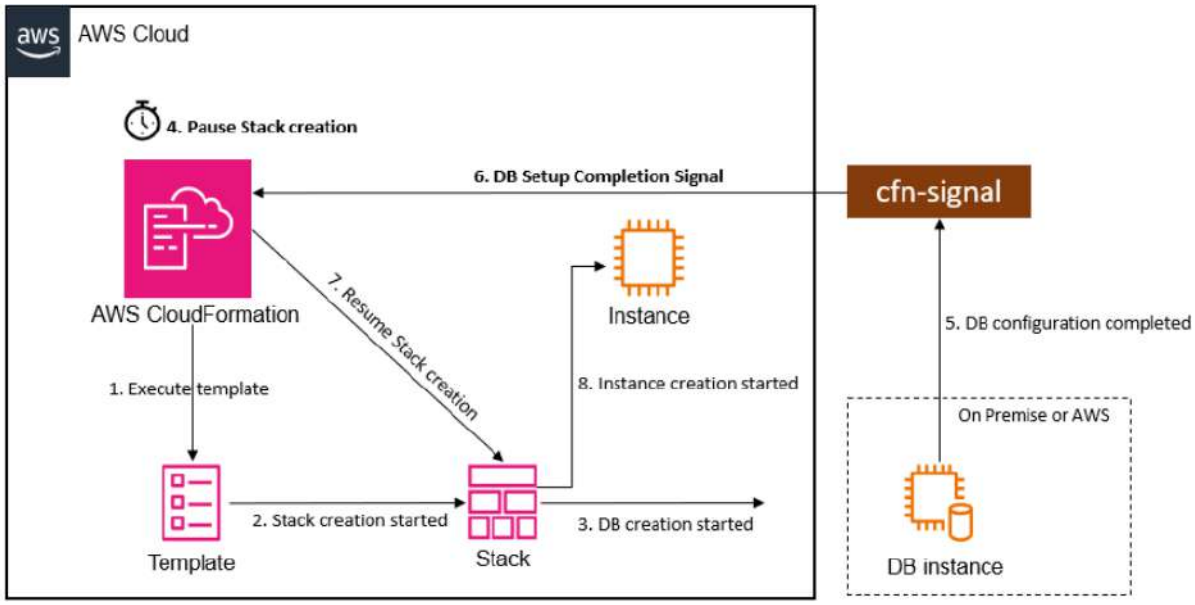


▼ ☑ **CloudFormation Policy Attributes.**

CreationPolicy attribute allows us to specify a policy for how CloudFormation should handle the creation of a resource. This is often used with resources that need additional time to become fully functional or require validation before they are considered created. We can prevent atrribute status from reaching create complete until AWS CloudFormation receives a specified number of success signals or the timeout period is exceeded. To signal a resource, we can use the cfn-signal helper script or SignalResource API.

```
Resources:
  MyInstance:
    Type: AWS::EC2::Instance
    CreationPolicy:
      ResourceSignal:
```

```
        Count: 1
        Timeout: PT15M
```



DependsOn attribute specifies that the creation of one resource depends on the creation of another resource. This is useful when you need to control the order in which resources are created or updated.

```
Resources:
  MyBucket:
    Type: AWS::S3::Bucket

  MyBucketPolicy:
    Type: AWS::S3::BucketPolicy
    DependsOn: MyBucket
```

UpdatePolicy attribute defines how CloudFormation should handle updates to a resource. This is useful for managing rolling updates or handling update failures for certain resources.

```
Resources:
  MyAutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
```

```
    UpdatePolicy:
      AutoScalingRollingUpdate:
        MinInstancesInService: 1
        MaxBatchSize: 1
        PauseTime: PT5M
```

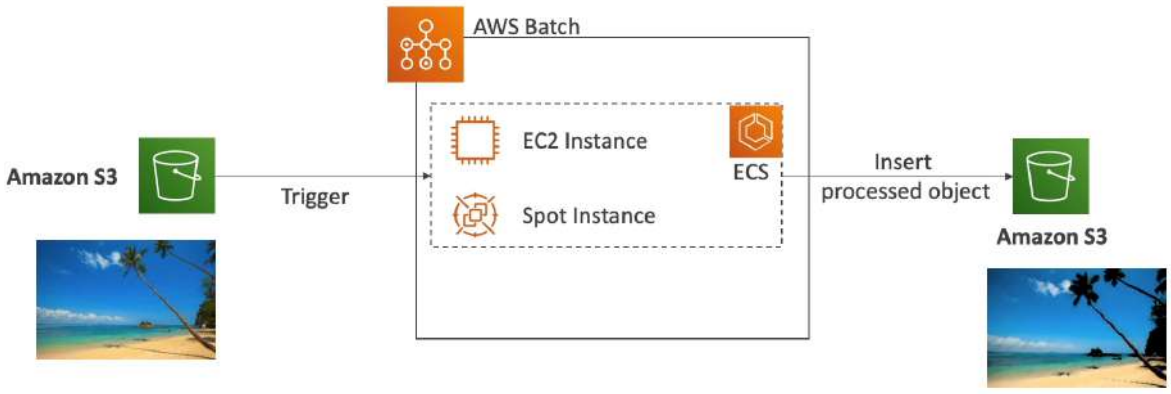
UpdateReplacePolicy attribute specifies what CloudFormation should do with a resource when it is replaced during an update. This is useful for controlling the behavior of resources like RDS instances or S3 buckets when they are replaced.

```
Resources:
  MyBucket:
    Type: AWS::S3::Bucket
    UpdateReplacePolicy: Retain
```

▼ ☑ **AWS Batch.**

AWS Batch is a fully managed batch processing service at any scale (efficiently runs 100 000s of computing batch jobs on AWS). A batch job is a job with a start and an end.

Batch jobs are defined as Docker images and run on ECS. Batch will dynamically launch EC2 instances or Spot Instances and will provision the right amount of compute/memory.



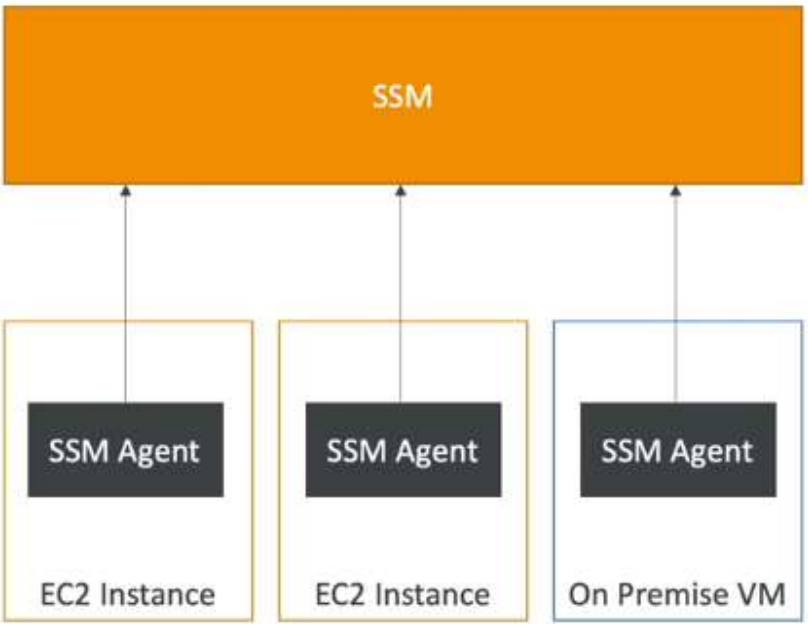
▼ ☑ **AWS SSM.**

SSM is a hybrid service that helps us manage our EC2 and On-Premises systems at scale. We can get operational insights about the state of our infrastructure.

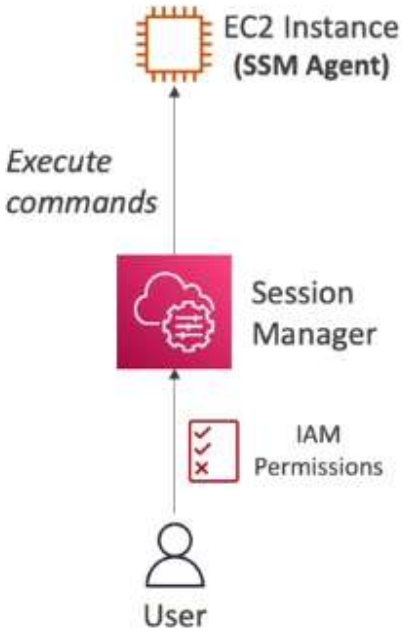
Most important features of SSM:

- Patching automation for enhanced compliance.
- Run commands across an entire fleet of servers.
- Store parameter configuration with the SSM Parameter Store.

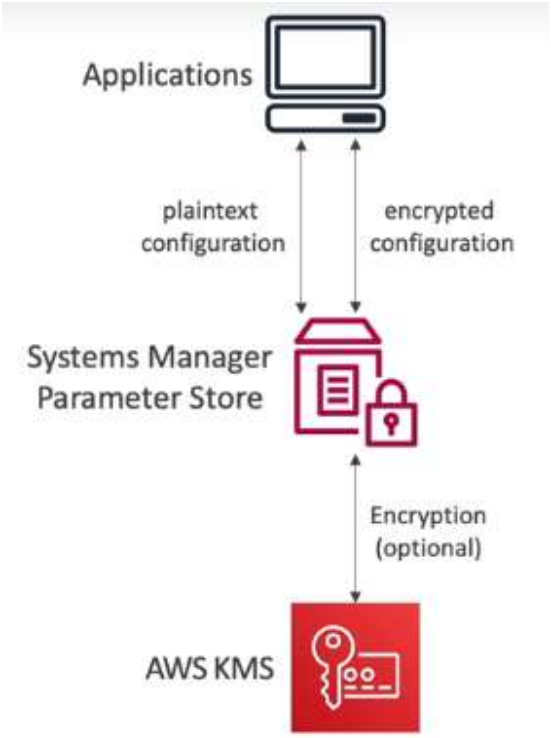
We need to install the SSM agent onto the systems we control (installed by default on Amazon Linux AMI). If an instance cant be controlled with SSM, it's probably an issue with the SSM agent. With SSM agent we can run commands, patch and configure our servers.



SSM Session Manager - allows us to start SSH on our EC2 and on-premises servers. No SSH access, bastion hosts or SSH keys needed. Also if we want better security we can disable port 22.



SSM Parameter Store - it is a secure storage for configuration and secrets (API keys, passwords, configurations ...). It is serverless, scalable and durable. Also it is secure because we control access permissions using IAM.

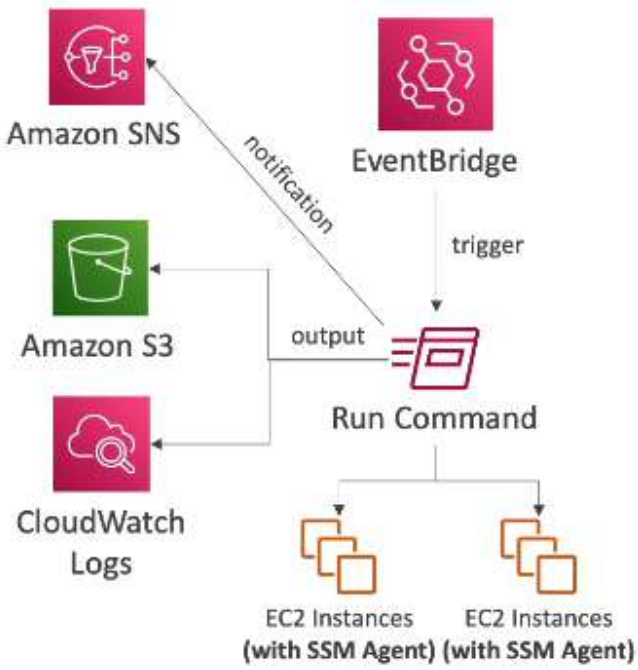


▼ ☐ SSM Advanced.

- **SSM Run Command**

  It lets us to execute a script or just run a command. We can run command across multiple instances (using resource groups) and we dont need SSH. It can be integrated with IAM & CloudTrail and can be invoked using EventBridge.

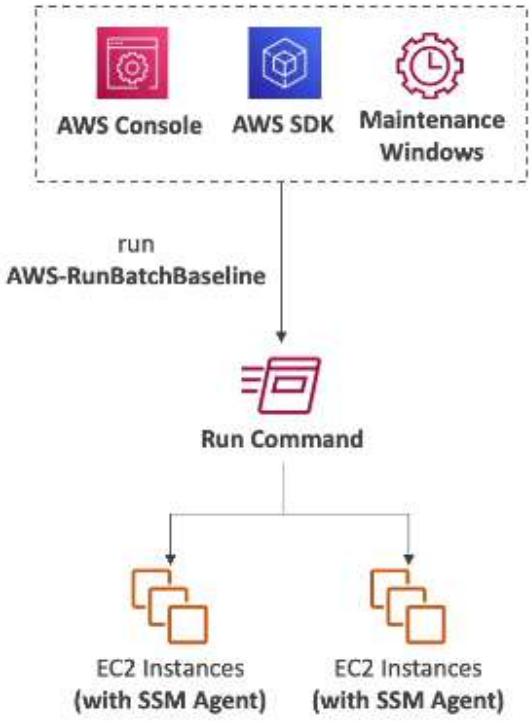
  Command output can be shown in the AWS Console, sent to S3 bucket or CloudWatch Logs. We can also send notifications to SNS about command status.



- **SSM Patch Manger**

  It automates the process of patching managed instances (OS updates, applications updates, security updates). It supports EC2 instances and on-premises servers. Can scan instances and generate patch compliance.

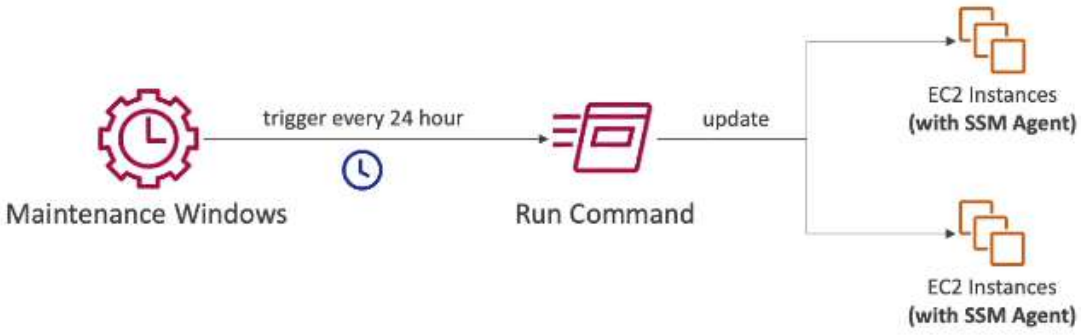
  We can patch on-demand or on a schedule using Maintenance Windows.



- **SSM Maintenance Windows**

  It defines a schedule for when to perform actions on our instances (e.g. OS patching, updating drivers, installing software).

  Maintenance Windows contains: shcedule, duration, set of registered instances and set of registered tasks.



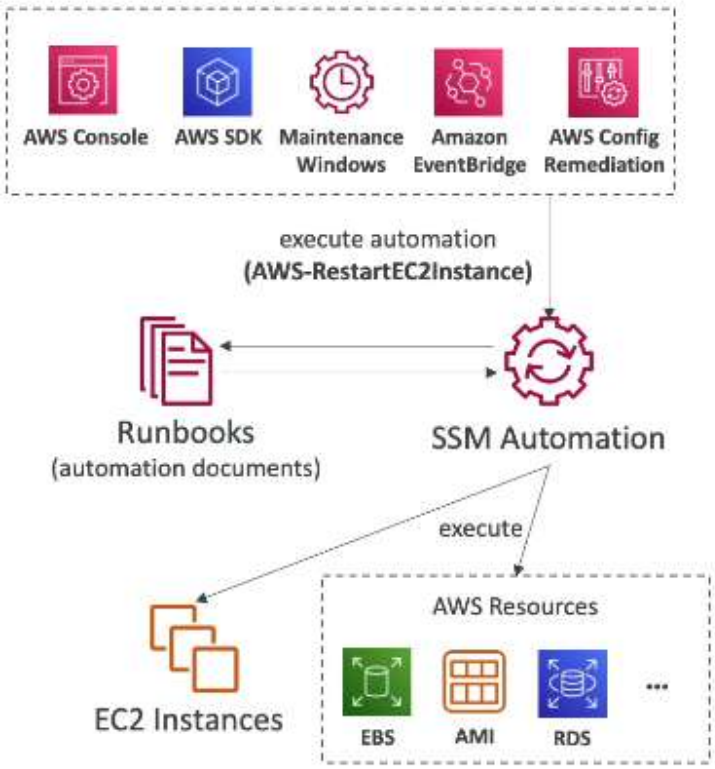
- **SSM Automation**

  It simplifies common maintenance and deployment tasks of EC2 instances and other AWS resources (e.g. restart instances, create an AMI, EBS snapshot).

Automation Runbook - define actions performed on our EC2 instances or AWS resources.

Automation can be triggered using:

- AWS Console, AWS CLI or SDK
- EventBridge
- Using Maintenance Windows
- AWS Config



▼ ☑ **AWS Elastic Beanstalk.**

Elastic Beanstalk is a **PaaS** offered by AWS. It is developer-centric view of deploying an application on AWS (**web and non web**). It abstracts much of the infrastructure management, allowing developers to focus on writing code (we still have full control over the configuration) and developing features rather than dealing with the underlying infrastructure. Only the application code is the responsibility of the developer. Can be used for deploying Docker containers.

Beanstalk is free but we need to pay for the underlying instances.

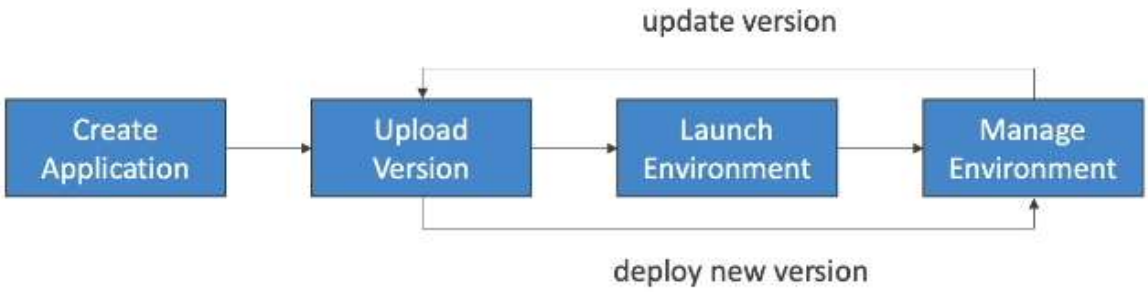
Elastic Beanstalk architecture models:

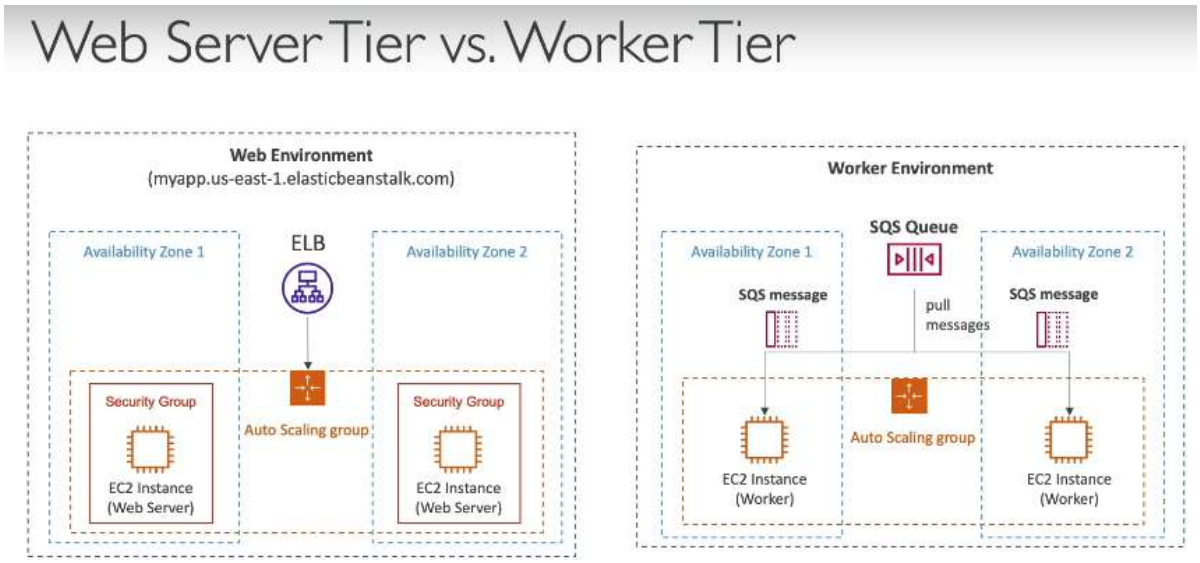
- **Single Instance deployment** - good for development.
- **LB + ASG** - great for production or pre-production web apps.
- **ASG only** - great for non-web apps in production.

Health Monitoring - a crucial feature that helps ensure the reliability and availability of our applications. It involves tracking the status of the environment's resources and application instances, detecting issues, and taking appropriate actions to maintain the desired state of the environment. It pushes metrics to CloudWatch.

Elastic Beanstalk Components

- Application - collection of Elastic Beanstalk components (environments, versions, configurations, ...).
- Application Version - an iteration of our application code.
- Environment - collection of AWS resources running an application version (only one application version at a time).





▼ ☑ **AWS Amplify.**

AWS Amplify is a set of tools and services designed to help developers build scalable and full-stack applications on AWS. It provides a straightforward and comprehensive way to develop web and mobile applications, leveraging the power of AWS cloud services.

▼ ☑ **Amazon SES.**

SES (Simple Email Service) is a fully managed service used to send email securely, globally and at scale. It allows inbound and outbound emails. We can send emails using our application, AWS Console, APIs or SMTP.

We have access to reputation dashboard, performance insights and anti-spam feedback. It provides statistics such as email deliveries, bounces, feedback loop results, email open, etc...

It supports DomainKeys Identified Mail (DKIM) and Sender Policy Framework (SPF).

Use cases: transactional, marketing and bulk email communications.

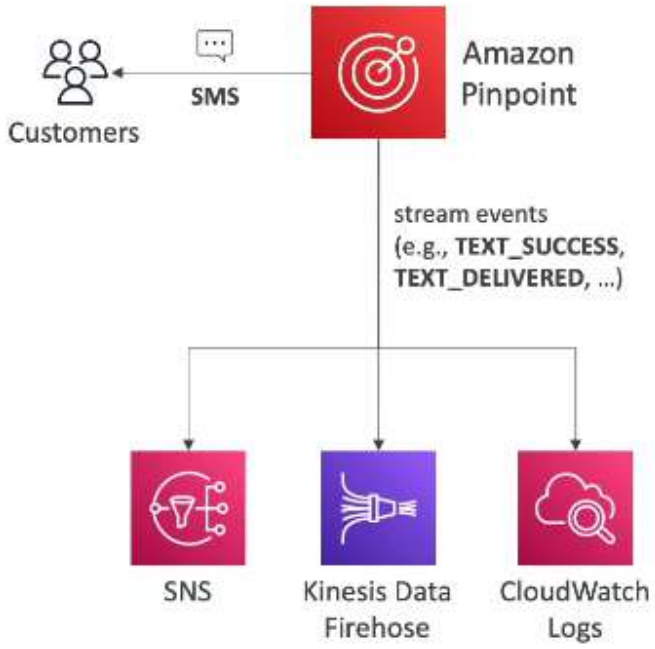
▼ ☑ **Amazon Pinpoint.**

Pinpoint is a scalable 2-way (outbound/inbound) marketing communications service. It supports email, SMS, push, voice and in-app messaging and has ability to segment and personalize messages with the right content to customers. Also we have possibility to receive replies.

Use cases: run campaigns by sending marketing, bulk, transactional SMS messages.

Pinpoint vs SNS & SES

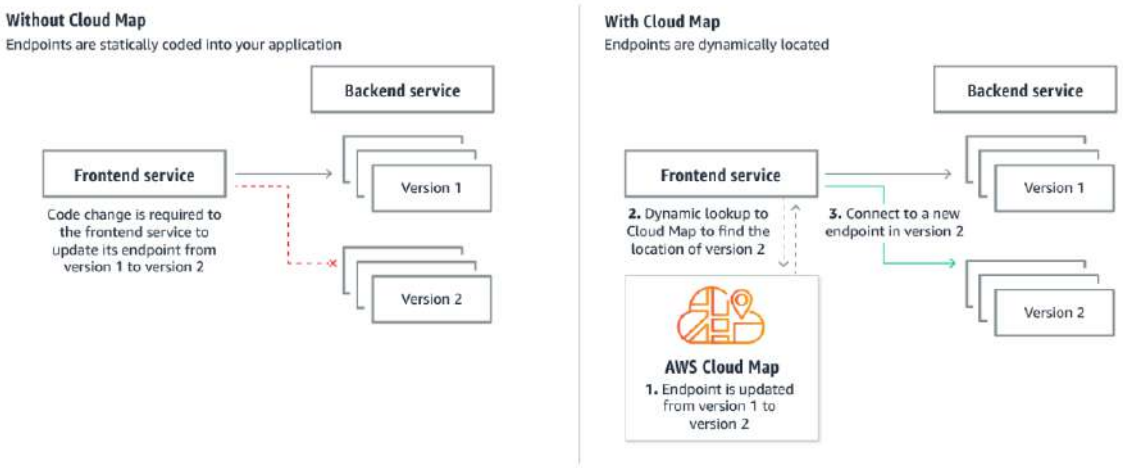
- In SNS & SES we manage each message's audience, content and delivery schedule.
- SQS focuses on reliable message queuing and decoupling applications.
- SES focuses on reliable email delivery and managing email reputation.
- In Pinpoint we create message templates, delivery schedules, highly-targeted segments and full campaigns. Focuses on customer engagement and analytics.



▼ ☑ **AWS Cloud Map.**

Cloud Map is a service that allows us to easily create and manage custom names for our application resources. It simplifies the process of service discovery in cloud-based applications by maintaining a dynamic directory of all our services and their locations.

It constantly monitors the health of every IP-based component of our application and dynamically updates the location of each microservice as it is added or removed.



▼ ☑ **AWS Cost Anomaly Detection.**

Cost Anomaly Detection continously monitor our cost and usage using ML to detect unsual spends. It learns our unique, historic spend patterns to detect one-time cost spike and/or continous cost increases (we dont need to define tresholds).

We can monitor AWS services, member accounts, cost allocation tags or cost categories. It will send us anomaly detection report with root-cause analysis and we can get notified with alerts using SNS.

▼ ☑ **AWS Cost Explorer.**

Cost Explorer - allows users to visualize, understand, and manage their AWS costs and usage over time. We can create custom reports that analyze cost and usage data on high level (total cost and usage across all accounts).

With Cost Explorer we are able to access our optimal savings plan. Also we can forecast usage up to 12 months based on previous usage.

▼ ☑ **AWS Trusted Advisor.**

Trusted Advisor is a service that helps customers optimize their AWS environments, improve performance, and save money. It provides real-time guidance to help users follow AWS best practices.

It lets us analyze our AWS accounts and provides recommendation on six categories:

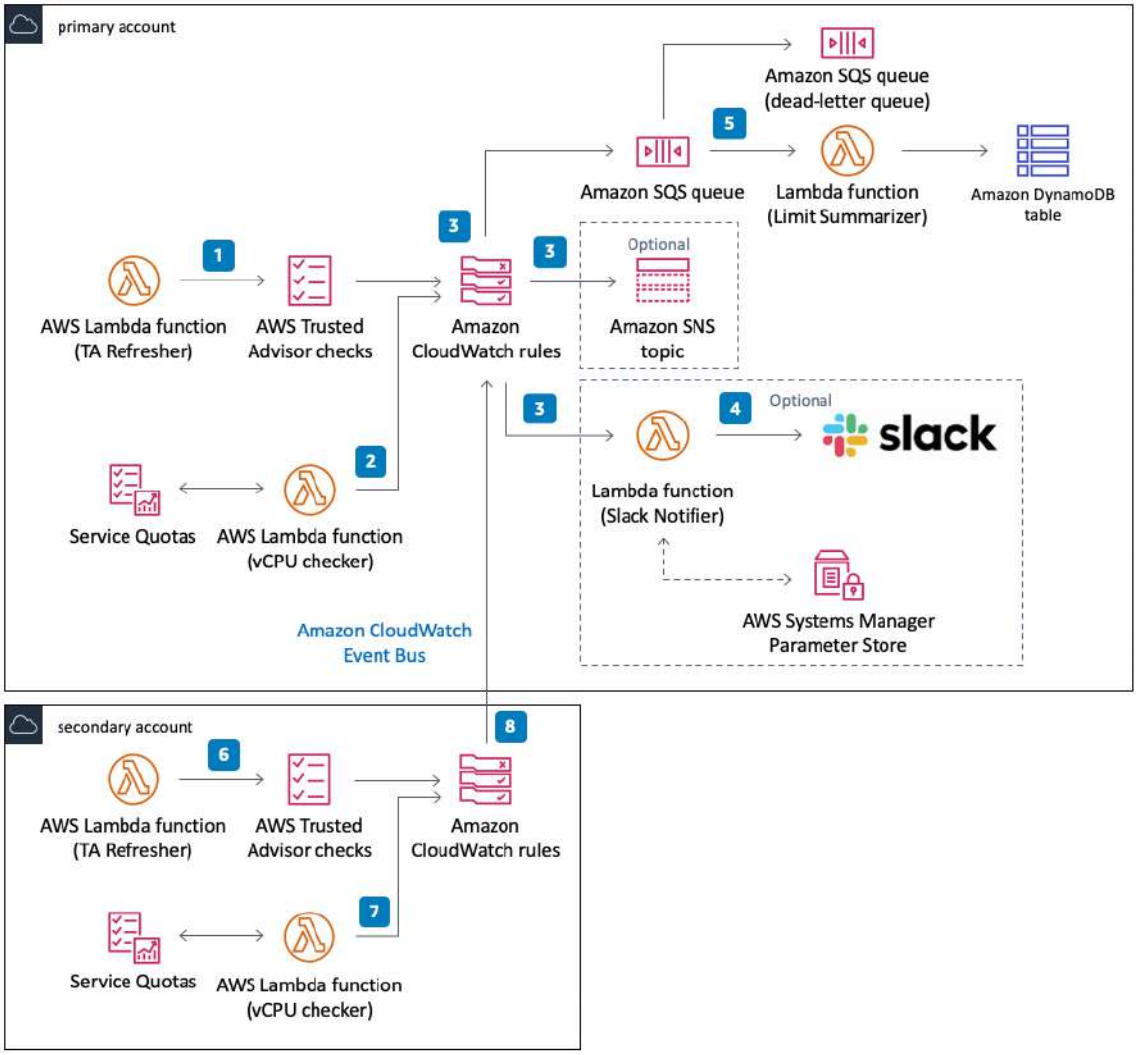
- Cost optimization.
- Performance.
- Security.
- Fault tolerance.
- Service limits.
- Operational Excellence.

For Business & Enterprise Support plan:

- Full Set of Checks.
- Programmatic Access using AWS Support API.

▼ **Solution Architecture - Quota Monitor.**

Pattern uses an AWS Lambda function that runs once every 24 hours. The Lambda function refreshes the AWS Trusted Advisor Service Limits checks to retrieve the most current utilization and quota data through API calls. Amazon CloudWatch Events captures the status events from Trusted Advisor. It uses a set of CloudWatch Events rules to send the status events to all the targets we choose during initial deployment of the solution: SQS queue, SNS topic or a Lambda function for Slack notifications.



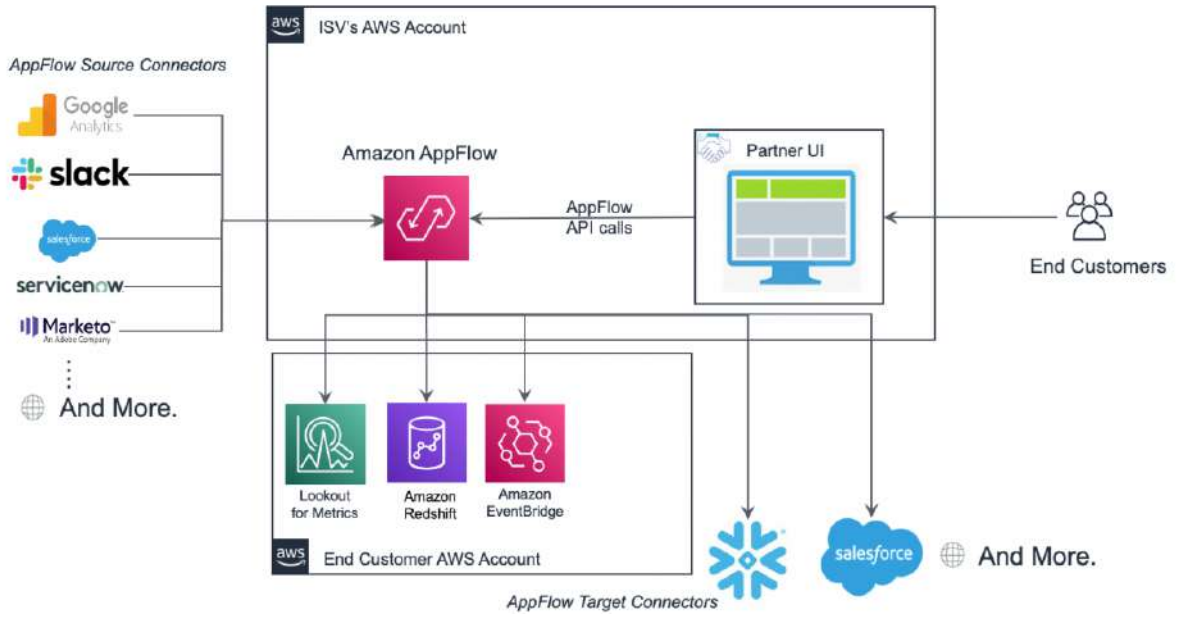
▼ ☑ **Amazon AppFlow.**

AppFlow is a fully managed integration service that enables us to securely transfer data between SaaS applications and AWS. Data is encrypted over the public internet or privately over AWS PrivateLink.

Sources: Salesfroce, SAP, Zendesk, Slack, ServiceNow...

Destinations: AWS Services (S3, Redshift) or non-AWS (SnowFlake, Salesforce).

We can define integration to run on a schedule, in response to events or on demand.



We dont spend time writing the integrations, we can leverage APIs immediately.

WhitePapers & Architectures

▼ ☑ **AWS Well Architected Framework.**

Well Architected Framework is a set of best practices designed to help cloud architects build secure, high-performing, resilient, and efficient infrastructure for their applications.

AWS Cloud Best Practices - Design Principles

- **Scalability** - vertical & horizontal.
- **Disposable Resources** - servers should be disposable & easily configured.
- **Automation** - serverless, IaaS, auto scaling ...
- **Loose Coupling**
- **Services, not Servers** - dont use just EC2, use managed services, databases, etc.

Well Architected Framework **Pillars** - key areas of focus that AWS recommends when designing and evaluating architectures on AWS.

▼ ☑ **WhitePapers Links.**

1. Architecting for the cloud:

https://d1.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf (Archived)

2. WhitePapers related to well-architected framework:

https://aws.amazon.com/blogs/aws/aws-well-architected-framework-updated-white-papers-tools-and-best-practices/

3. Disaster recovery whitepaper:

https://d1.awsstatic.com/whitepapers/aws-disaster-recovery.pdf

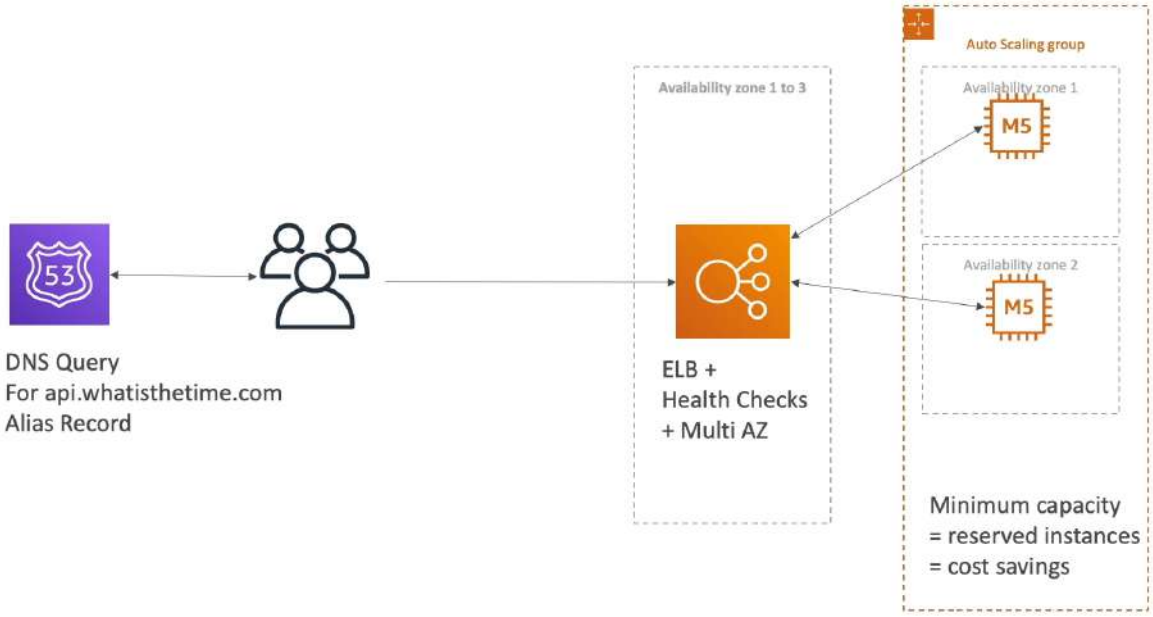
Well-Architected framework whitepaper:

https://d1.awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf

▼ **1. WhatsTheTime App**

Requirements:

- It allows people to know what time it is.
- We dont need a database.
- Start small and can accept downtime.
- We want to fully scale vertically and horizontally, no donwtime.



▼ **2. Clothes Store App.**

Requirements:

- It allows people to buy clothes online.
- There is a shopping cart and users should not lose their shopping cart.
- Website can handle hundreds of users at the same time.
- We need to scale, maintain horizontal scalability and keep our web application as stateless as possible.
- Users should have their details in a database.

Architecture

We can use basic architecture with Route 53, ELB (Multi-AZ) and ASG (3 AZ).

Session

We could use sticky session to save shoping cart but it will violate EC2 performance. Using web cookies is better but we have security risks. The best way to save session data is to store it in ElastiCache.

Database

We can store user data in RDS. Because there are hundreds of users we can use RDS Read Replicas to scale reads (there is alternative using Lazy Loading

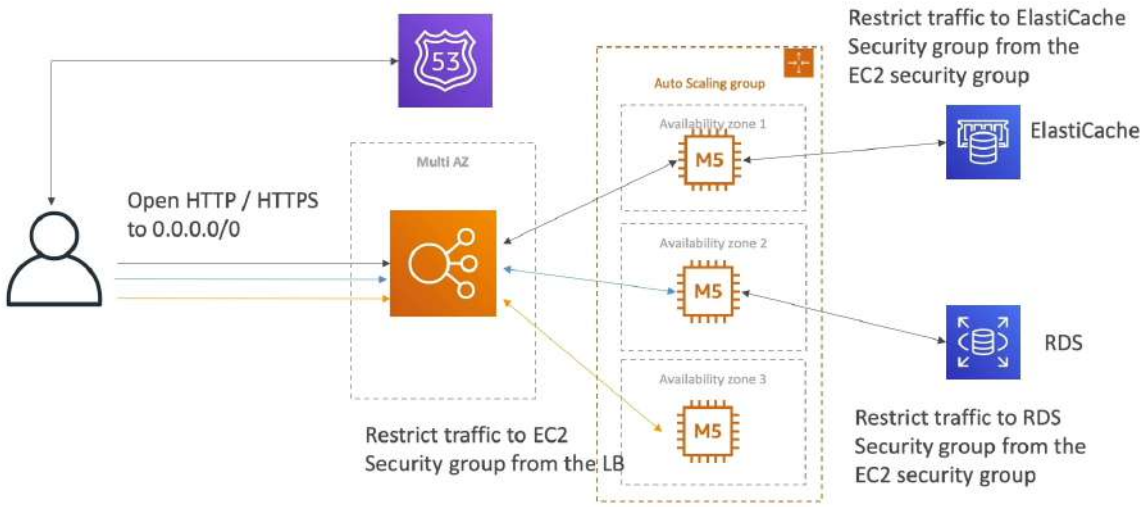
pattern with ElastiCache).

Availability

Using Multi-AZ RDS deployment can be used for disaster recovery.

Security

Open HTTP/HTTPS to 0.0.0.0/0 for users. We need to restrict traffic to EC2 security group from the load balancer and restrict traffic to ElastiCache and RDS security groups from the EC2 security group.

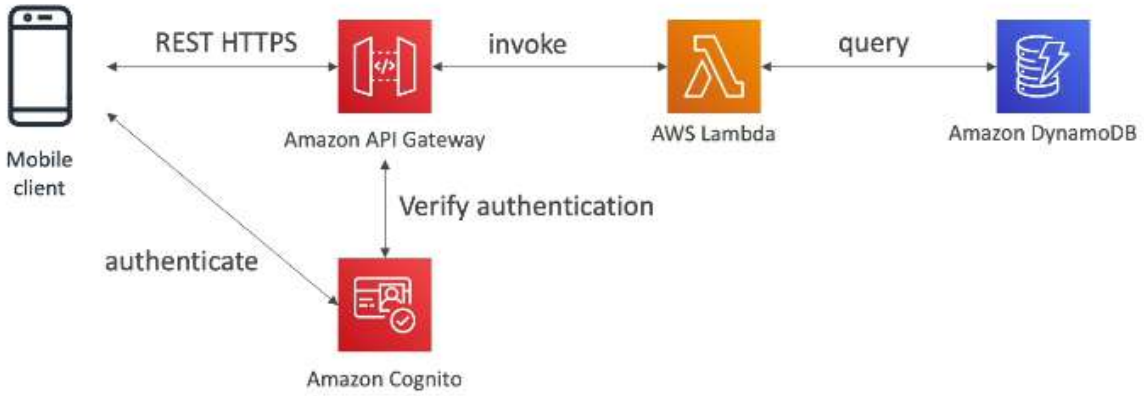


▼ **3. MyTodoList Mobile App.**

Requirements:

- Expose as REST API with HTTPS.
- Serverless architecture.
- Users should be able to directly interact with their own folder in S3.
- Users should authenticate through a managed serverless service.
- Users can write and read to-dos, but they mostly read them.
- The database should scale and have some high read throughput.

REST API Layer

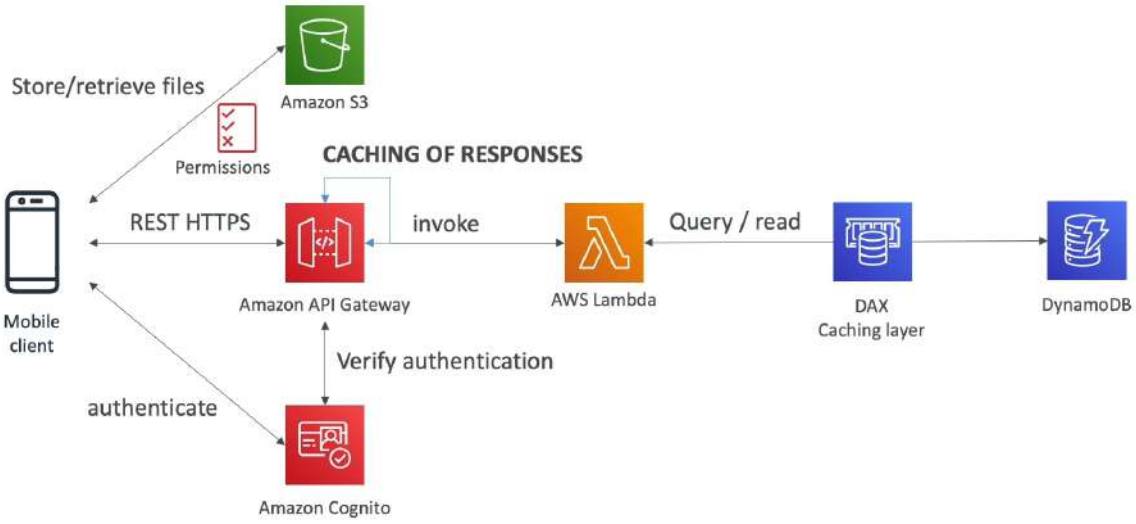


Giving Users Access to S3

We will use Cognito to generate us temporary credentials so we can use S3 to store and retrieve files.

High Read Throughput

We can use DAX as a caching layer, so reads will be cached and DynamoDB will not need many RCU. Another alternative is to cache responses at API Gateway level.



▼ 4. MyBlog Serverless Website.

Requirements:

- Website should scale globally.
- Blogs are rarely written, but often read.

- Some of the website is purely static files, the rest is a dynamic REST API.
- Caching must be implement where possible.
- Any new users that subscribes should receive a welcome email.
- Any photo uploaded to the blog should have a thumbnail generated.

Serving static content, globally and securely

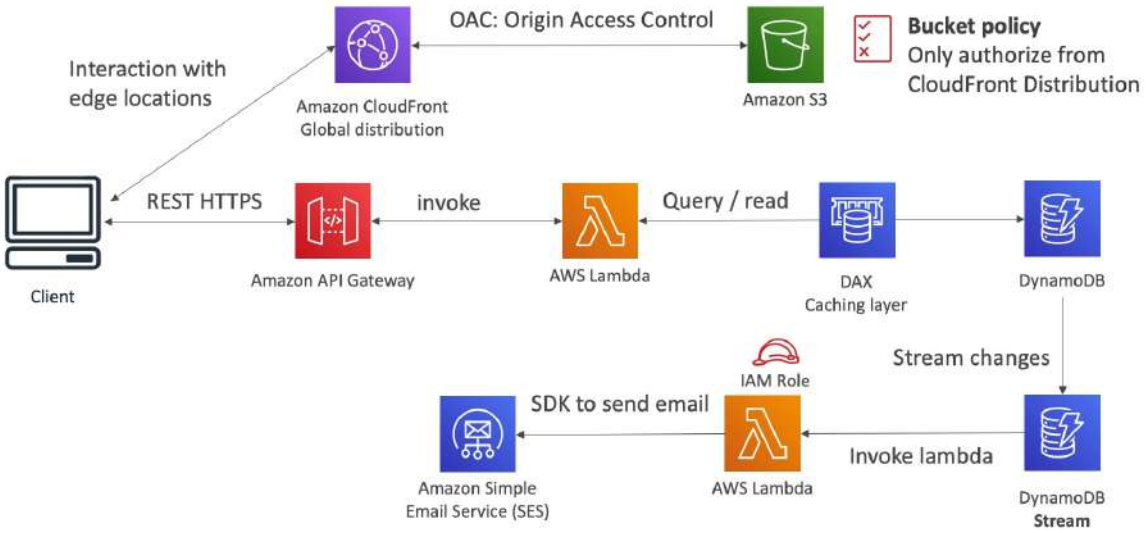
We can store our static content in S3 and we can use CloudFront to serve it globally. To secure it we will add Origin Access Control which will ensure that our S3 bucket can only be accessed by CloudFront. For this we will add a bucket policy to only auhorize the CloudFront distribution.

Serverless REST API

We will have REST HTTPS which will go through API Gateway, invoking a Lambda function and query/read from DynamoDB (can use DAX for cache). If we go global we can use DynamoDB Global Databases.

Welcome Email

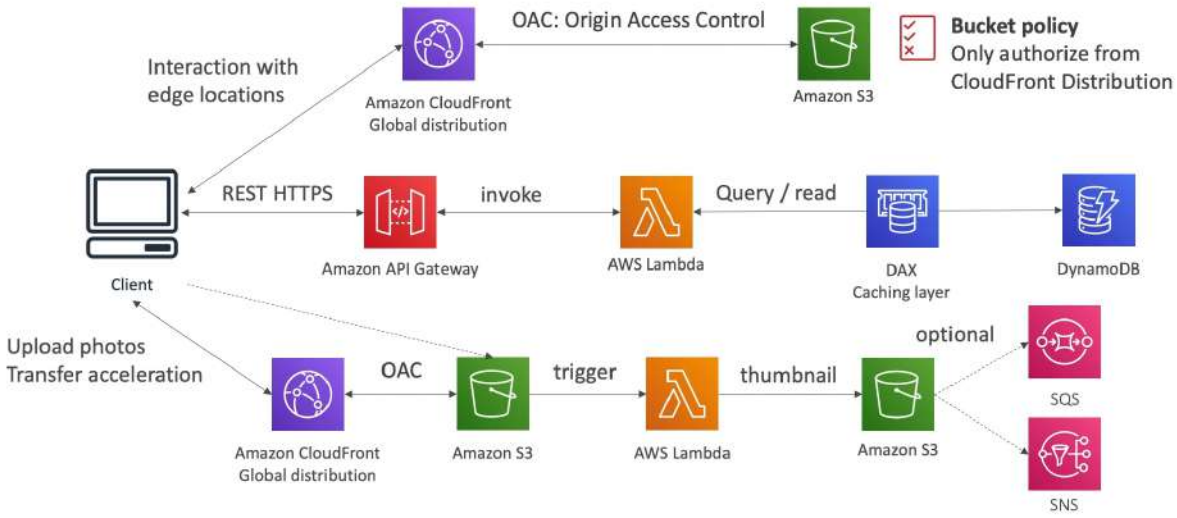
For user welcome email we can use DynamoDB Stream which will invoke a Lambda function (need IAM role) which will trigger SES to send an email.



Thumbnail Generation

When image is uploaded we need to store it in S3 bucket (can use again OAC and CloudFront - Transfer Acceleration). Uploading to S3 will trigger Lambda

which will create thumbnail and store it in another or same S3 bucket.

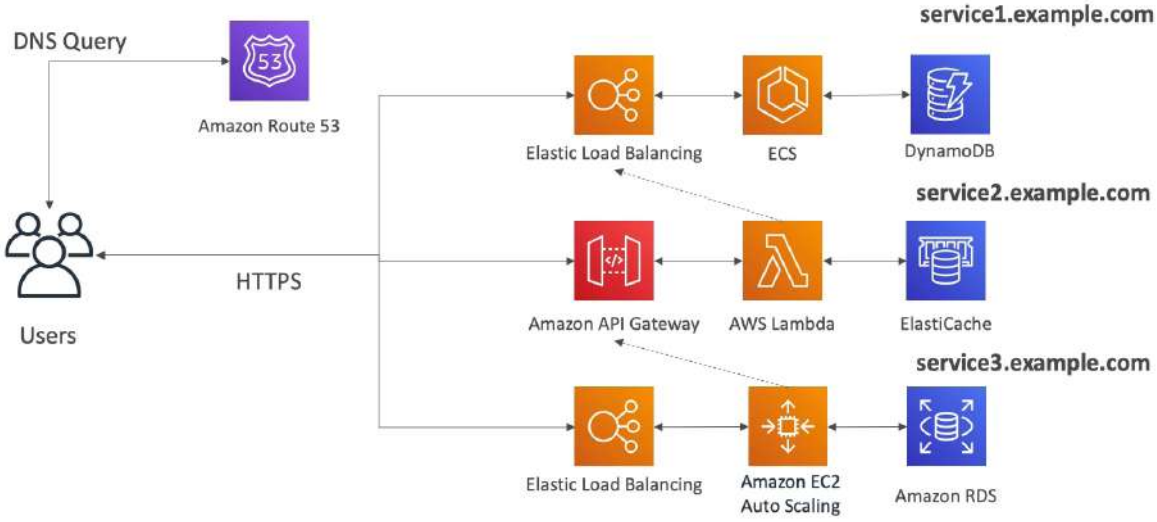


▼ 5. Microservices Architecture.

We are free to design each microservice the way we want.

Synchronous patterns: API Gateway, Load Balancers.

Asynchronous patterns: SQS, Kinesis, SNS, Lambda.



Challenges with microservices:

- Repeated overhead for creating each new microservice.
- Issues with optimizing server density/utilization.

- Complexity of running multiple versions of multiple microservices simultaneously.

Challenges solved by serverless patterns:

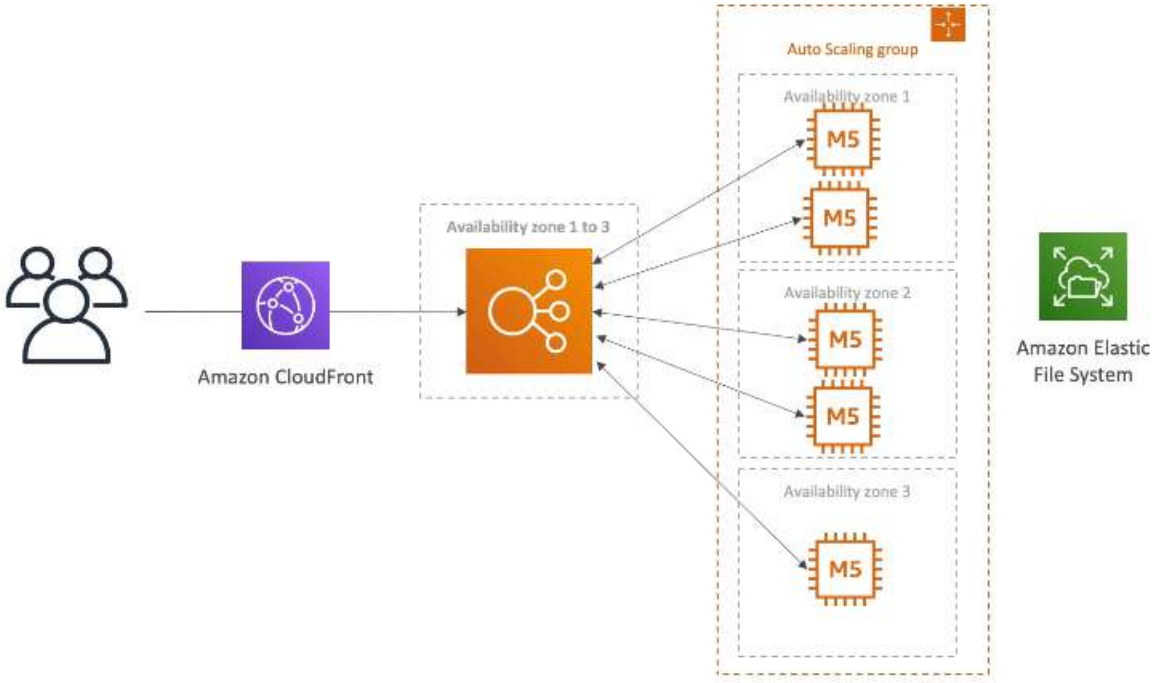
- API Gateway, Lambda scale automatically and we can pay per usage.
- We can easily clone API and reproduce enviornments.

▼ **6.** **Software Updates Distribution.**

We have an application running on EC2 that distributes software updates once in a while. When a new software update is out, we get a lot of requests and the content is distributed in mass over the network and it's very costly.

We can use CloudFront. It will cache software update files at the edge and no changes are needed to the architecture. Software update files are not dynamic, they are static (never changing).

EC2 isntances arent serverless, but CloudFront is and it will scale for us. We will save in availability, network bandwidth cost, etc.



▼ **7.** **Big Data Ingestion Pipeline.**

Requirements:

- We want the ingestion pipeline to be fully serverless.

- We want to collect data in real time and transform that data.
- We want to query the transformed data using SQL.
- The reports created using the queris should be in S3.
- Reported data should be loaded into a warehouse and we want to create dashboards.

Streaming

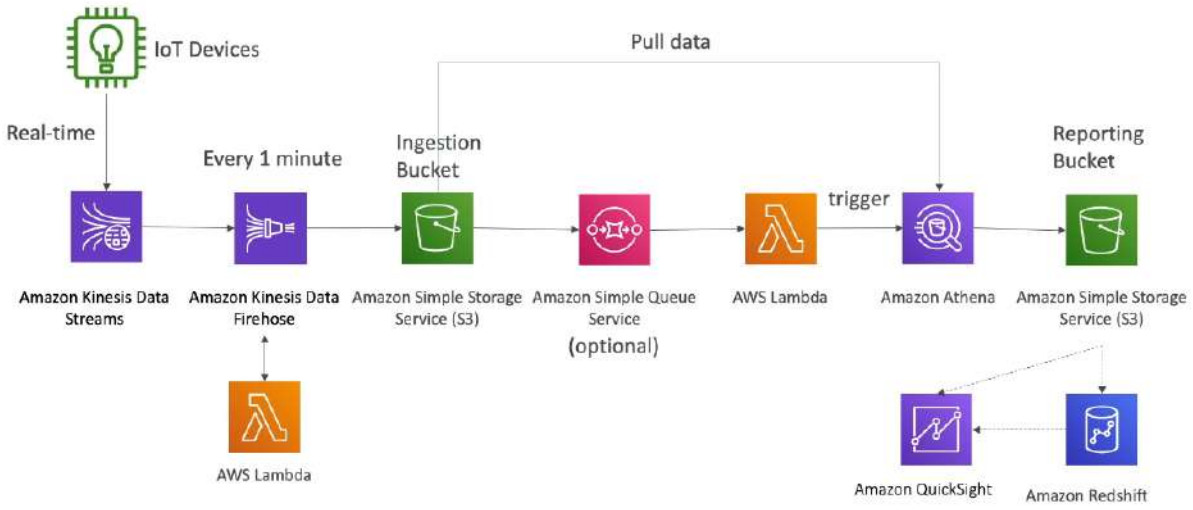
IoT devices send real-time data to Kinesis Data Streams. Data is periodically sent to Kinesis Data Firehose, which can trigger AWS Lambda for processing and then store it in S3 (Ingestion Bucket).

Processing

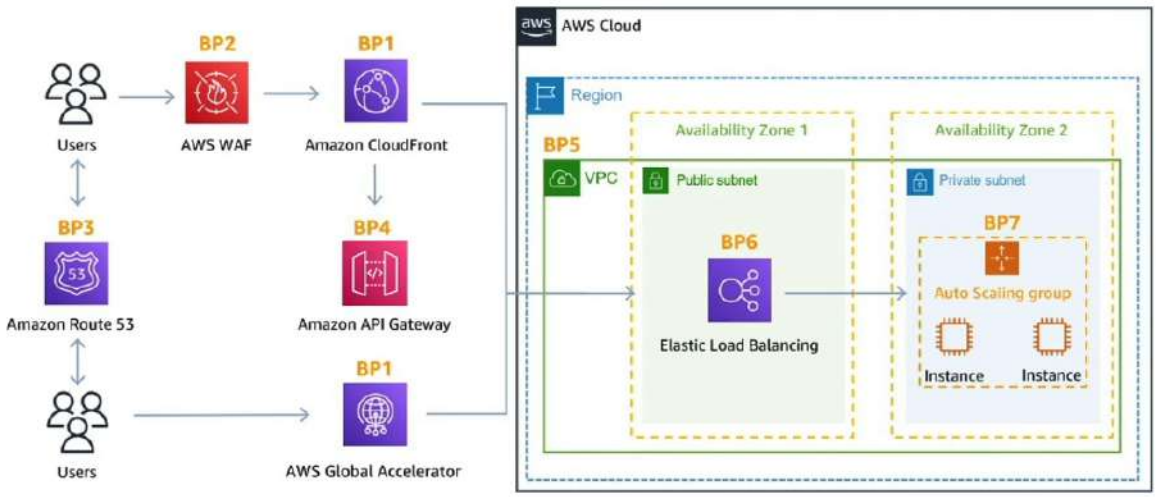
The data may go through SQS before further processing by another Lambda function. The processed data is analyzed using Amazon Athena, with results stored in the Reporting Bucket.

Analytics

The data in the Reporting Bucket or Redshift can be visualized using QuickSight or further analyzed in Redshift.



▼ 8. DDoS Protection Best Practices.



Edge Location Mitigation

BP1 - CloudFront ⇒ Web application delivery at the edge and we are protected from DDoS common attacks (SYN floods, UDP reflection...).

BP1 - Global Accelerator ⇒ Access our application from the edge. We can integrate it with Shield for DDoS protection.

BP3 - Route 53 ⇒ It has DDoS protection mechanism.

Best Practices for DDoS Mitigation

Infrastructure layer defense (BP1, BP3, BP6) ⇒ Protects EC2 against high traffic.

EC2 with ASG (BP7) ⇒ Helps in case of sudden traffic surges including a flash crowd or DDoS attack.

ELB (BP6) ⇒ Scales with the traffic increase and will distribute the traffic to many EC2 instances.

Application Layer Defense

Detect & filter malicious web requests (BP1, BP2) ⇒ CloudFront cache static content and serve it from the edge locations, protecting our backend. WAF is used on top of CloudFront and ALB to filter and block requests based on request signatures.

Shield Advanced (BP1, BP2, BP6) ⇒ Shield Advanced automatically creates, evaluates and deploys WAF rules to mitigate layer 7 attacks.

Attack Surface Reduction

Obfuscating AWS resources (BP1, BP4, BP6) ⇒ We can hide our backend resources (Lambda functions, EC2 instances)

Security groups and Network ACLs (BP5) ⇒ We can filter traffic based on specific IP at the subnet or ENI level. Elastic IP are protected by Shield Advanced.

Protecting API endpoints (BP4) ⇒ Hide our backend. If we use edge-optimized mode or CloudFront + regional mode we can get more control for DDoS protection. Also if we add WAF on top of it then we can get filtering of any HTTP request.

Other

▼ **Important Ports.**

Service	Port
FTP	21
SSH	22
SFTP	22
HTTP	80
HTTPS	443
PostgreSQL	5432
MySQL	3306
Oracle RDS	1521
MSSQL Server	1433
Remote Desktop	3389

▼ **Instantiating Applications Quickly.**

EC2 Instances

- Golden AMI - install our applications, OS dependecies, etc... beforehand and launch our EC2 instance from the Golden AMI.
- Bootstrap using User Data - for dynamic configuration.
- Hybrid - mix Golden AMI and User Data (Elastic Beanstalk).

RDS

- Restore from a snapshot - the database will have schemas and data ready.

EBS

- Restore from a snapshot - the disk will already be formatted and have data.