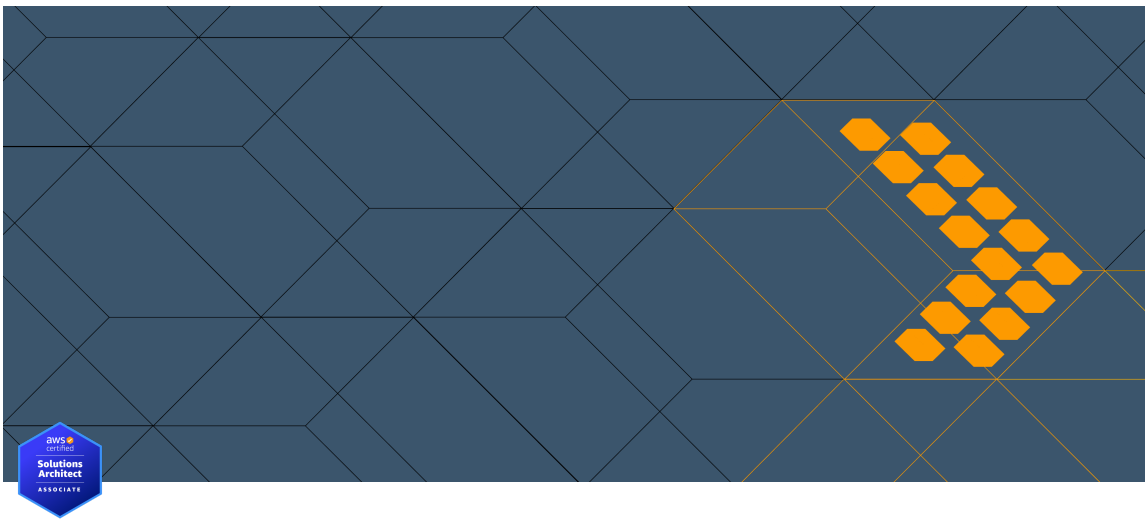
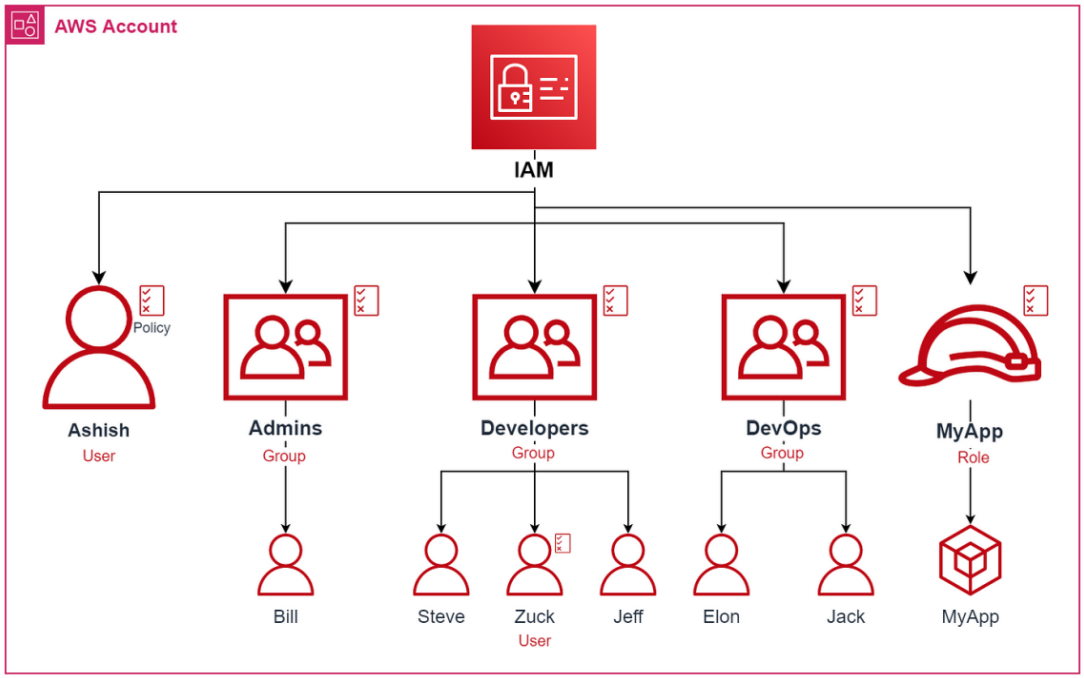


AWS SAA

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.**

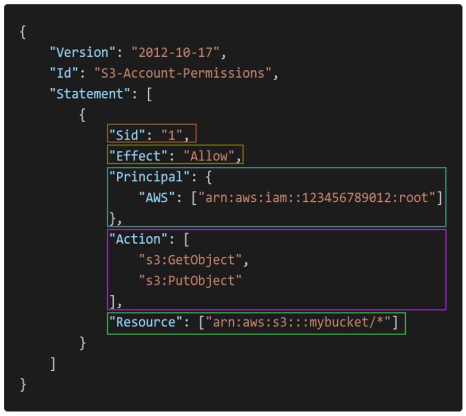
IAM Policy Components

- 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 Policies Structure

- Consists of
  - Version: policy language version, always include "2012-10-17"
  - Id: an identifier for the policy (optional)
  - Statement: one or more individual statements (required)
- Statements consists of
  - Sid: an identifier for the statement (optional)
  - Effect: whether the statement allows or denies access (Allow, Deny)
  - Principal: account/user/role to which this policy applied to
  - Action: list of actions this policy allows or denies
  - Resource: list of resources to which the actions applied to
  - Condition: conditions for when this policy is in effect (optional)



▼ ☑ 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)

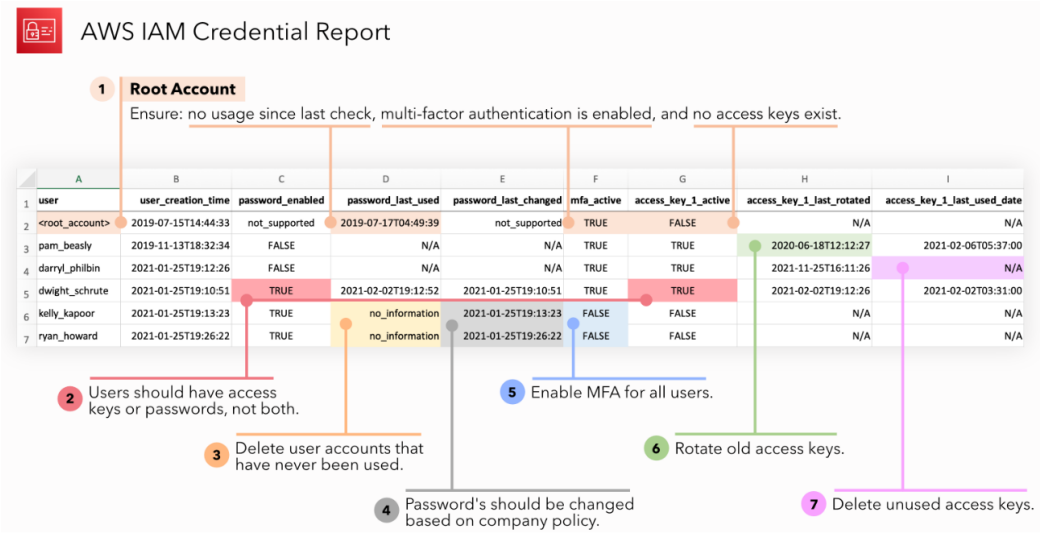
  It consists of libraries that let us to access and manage AWS services progammatically.

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.

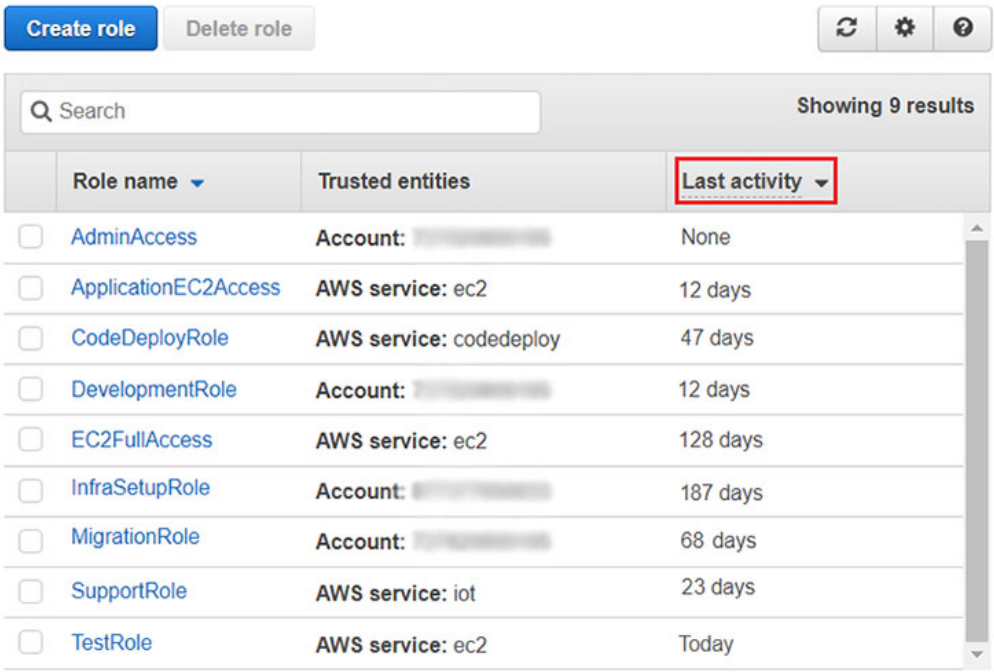
▼ ☑ **IAM Security Tools.**

- **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.

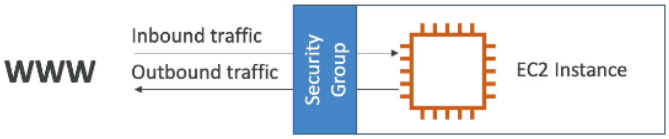
▼ ☑ **EC2 Instance Types.**

	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**

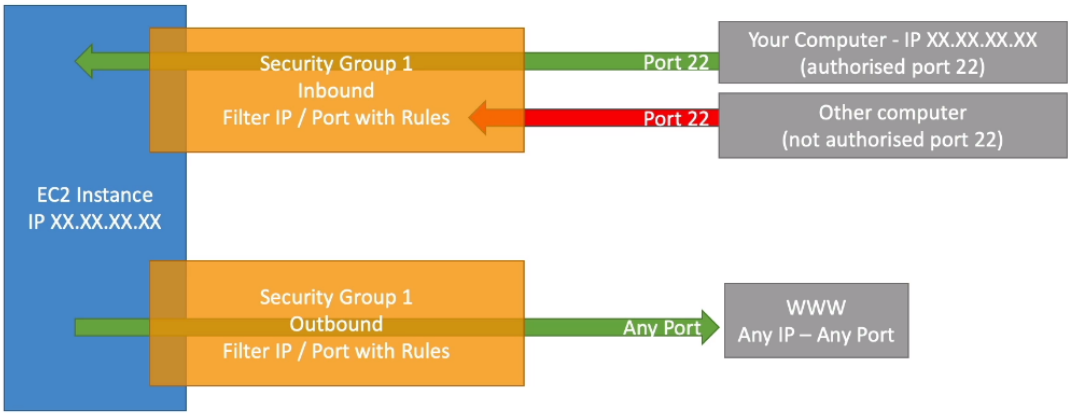
▼ ☑ **Security Groups.**

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.



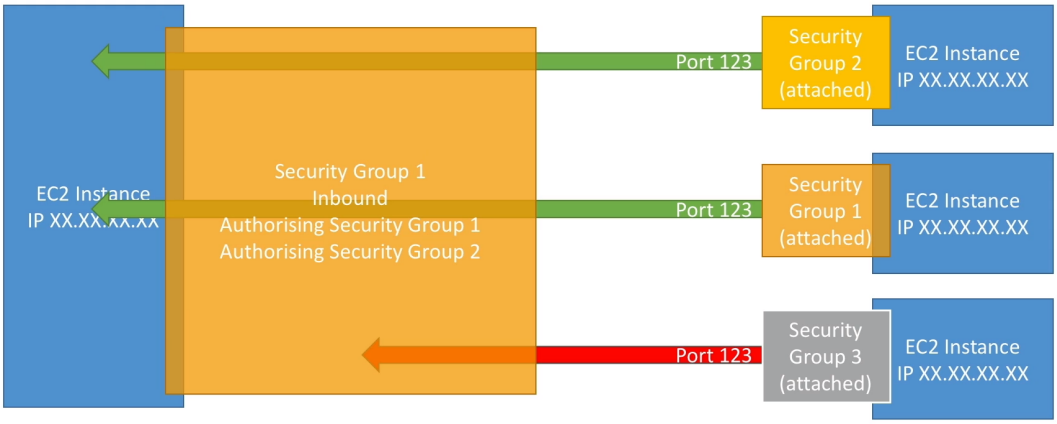
Security groups are acting as a "firewall" on EC2 instances. They regulate access to ports, authorised IP ranges, control of inbound network and control of outbound network.



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.**

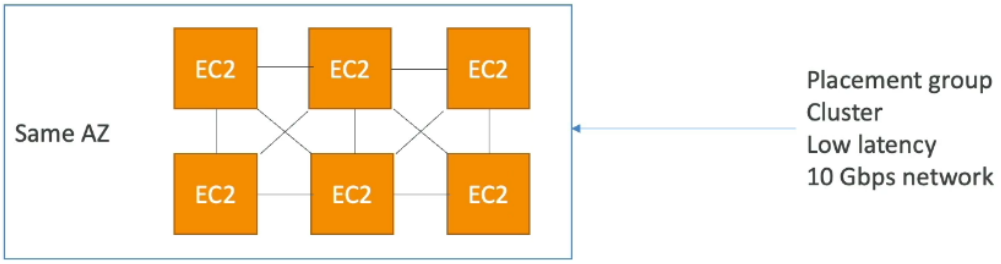
EC2 Placement Groups are a feature that enables us to influence the placement of Amazon EC2 instances on the underlying hardware. This can be important for workloads that require high performance, low latency, or other specific characteristics.

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.

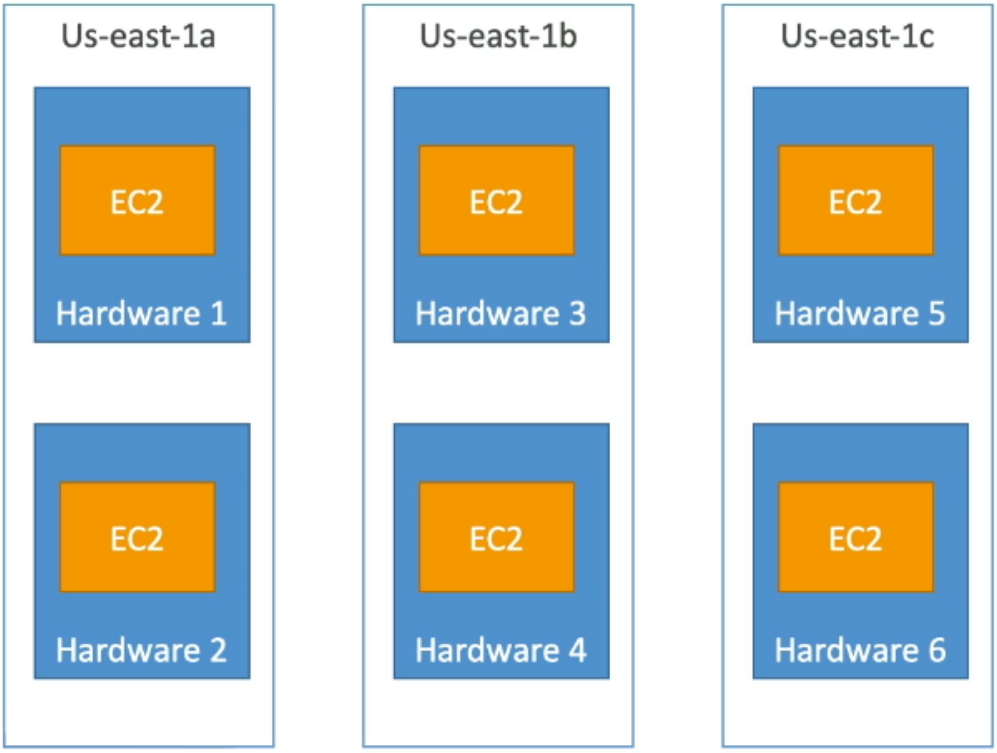
  All instances must be in the same AZ and there might be limits on the number of instances we can place in a single group depending on the instance type and available hardware.



- **Spread Placement Group**

  Suited for applications that require high availability.

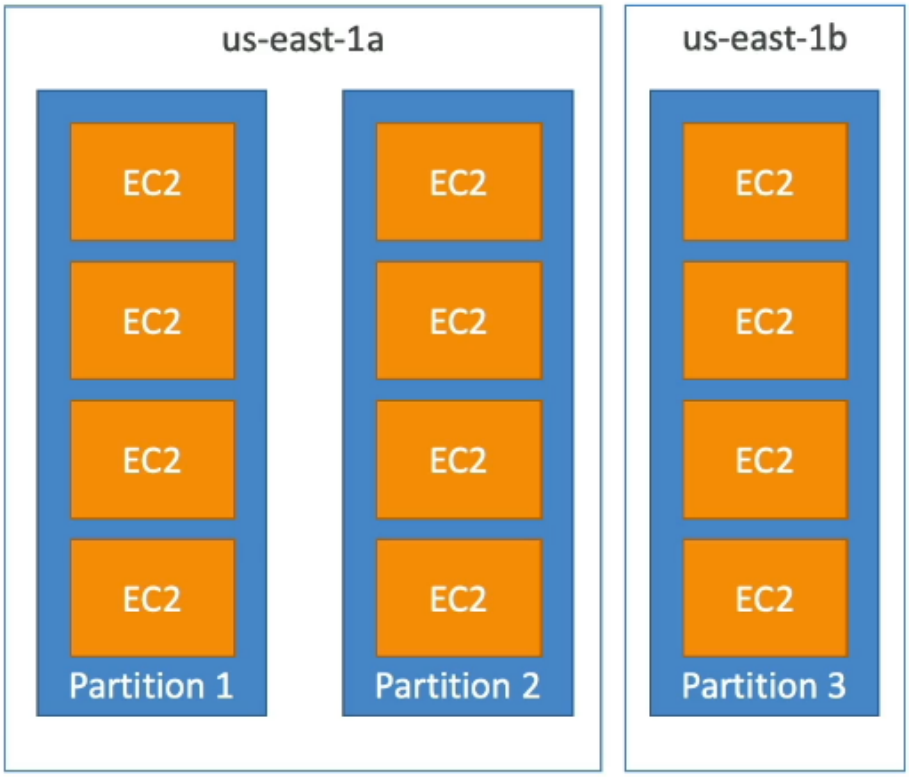
  Instances are distributed across distinct underlying hardware. This minimizes the risk of correlated hardware failures. Spread placement groups have stricter limits on the number of instances per AZ, typically up to seven.



- **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.



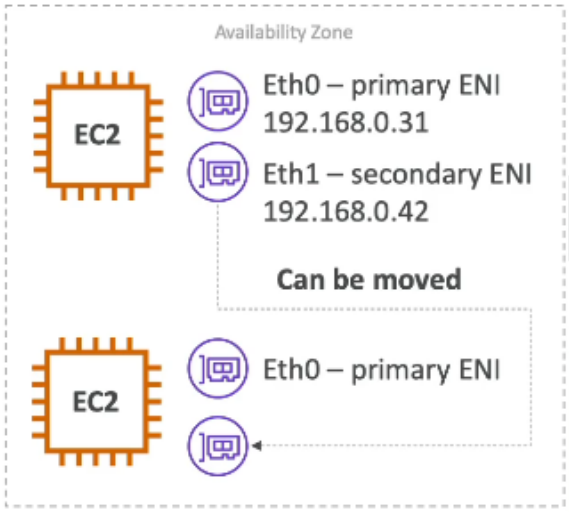
▼ ☑ **Elastic Network Interface (ENI).**

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**

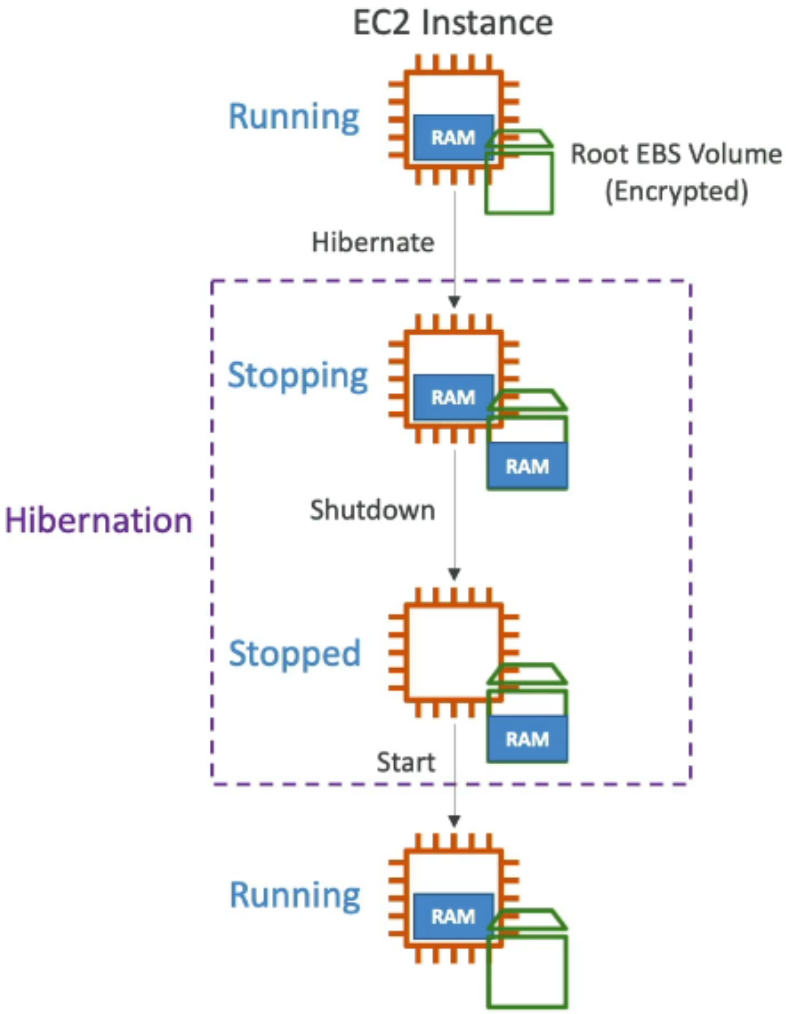
ENIs can be attached to instances in the same VPC, but must reside in the same AZ. We can attach a secondary ENI to an instance and use it for **failover** scenarios. If the primary instance fails, we can attach the ENI to a standby instance.



▼ ☑ **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.

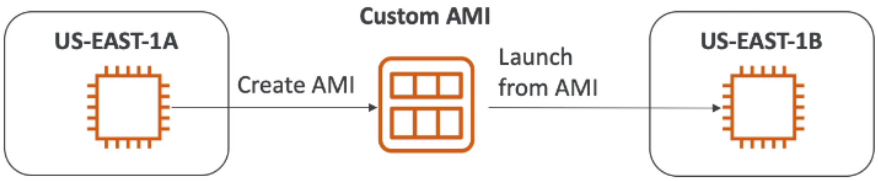
EC2 instances can be launched from:

- 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.

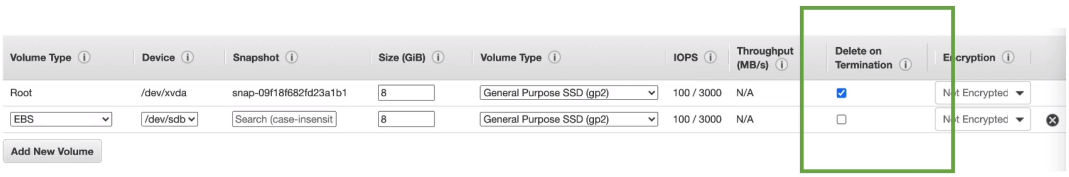
EBS can only be mounted to one instance at a time. They are bound to a specific AZ.

- 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.

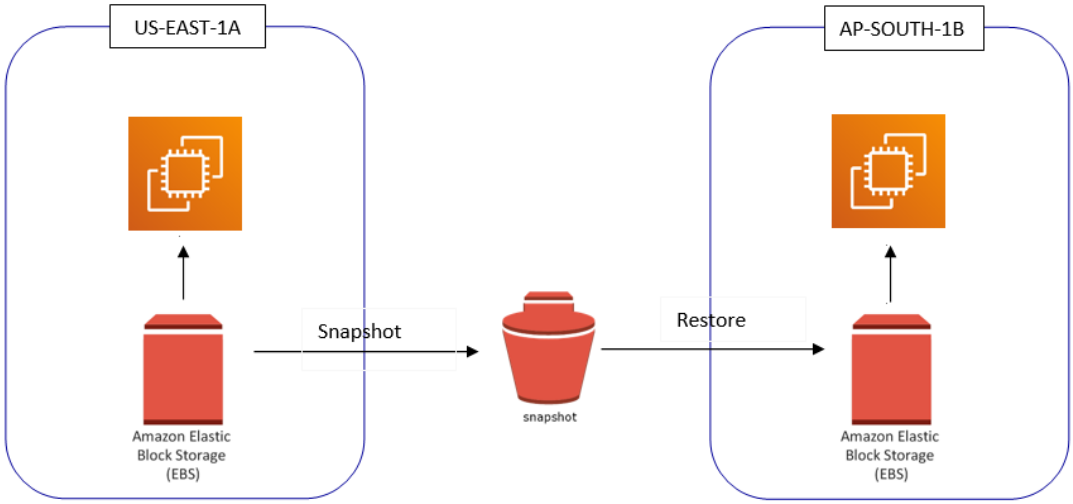
Delete on Termination

- 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.

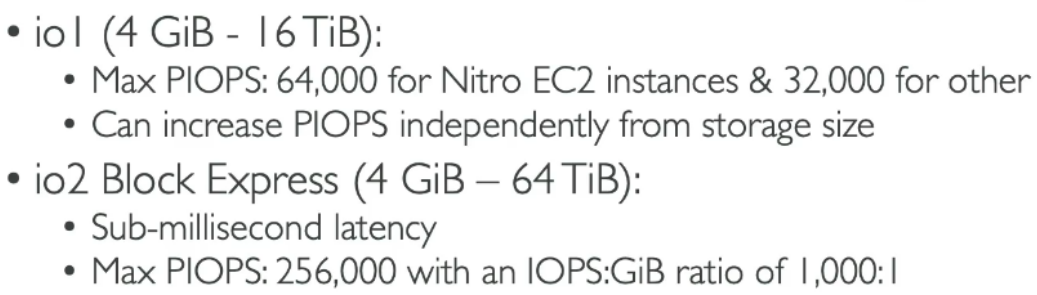
▼ ☑ **EBS Volume Types.**

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 in frequently 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.**

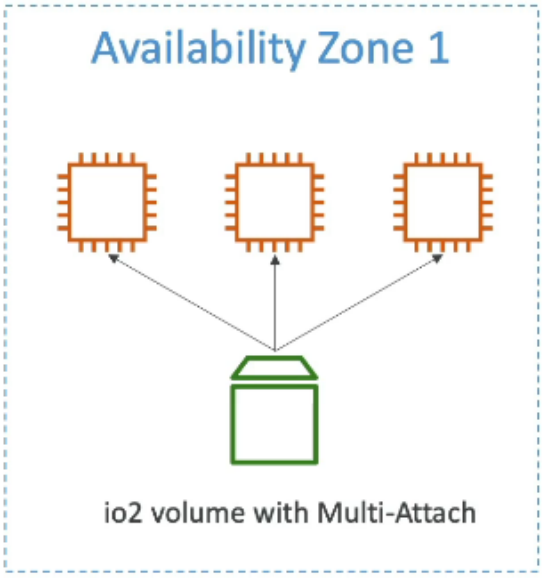
Multi-Attach lets us to attach same EBS volume to multiple EC2 instances in the same AZ. It is only possible with io1/io2 volume types. Each EC2 instance has full

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.

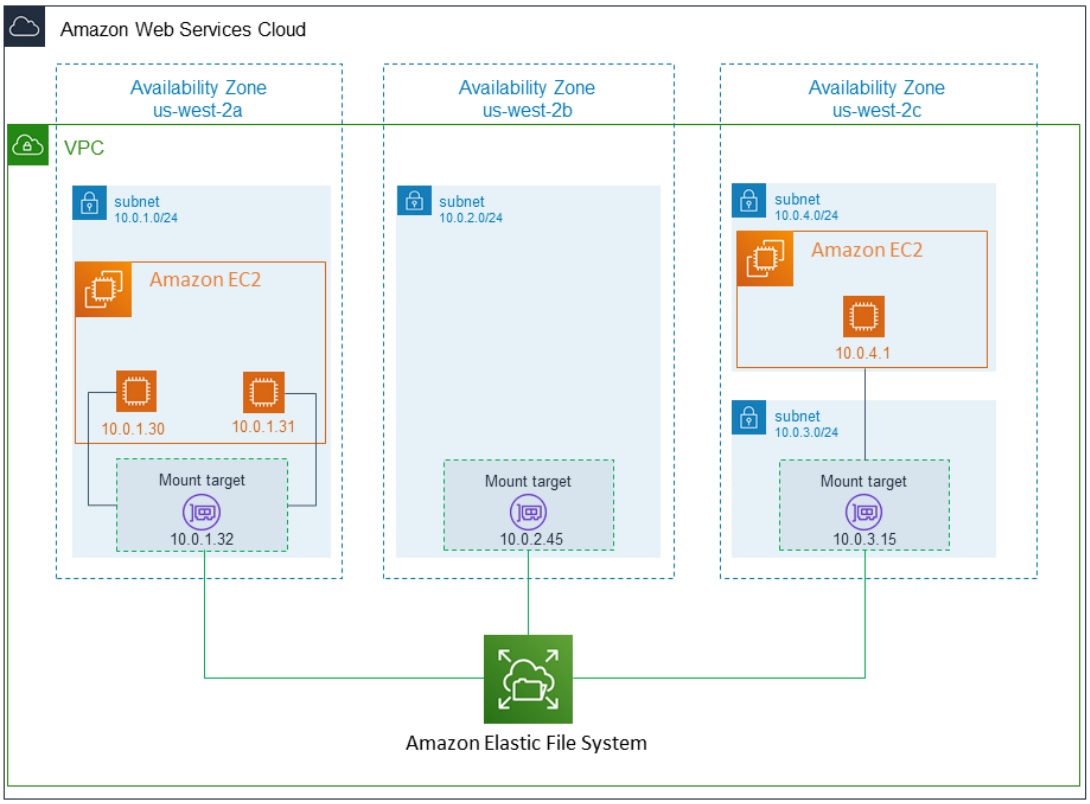
RAID Level	Performance	Redundancy	Capacity	Use Case
RAID 0	High	None	Full	Performance-critical, non-critical data
RAID 1	Moderate	Full	Half	Data reliability, critical systems

▼ ☑ **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

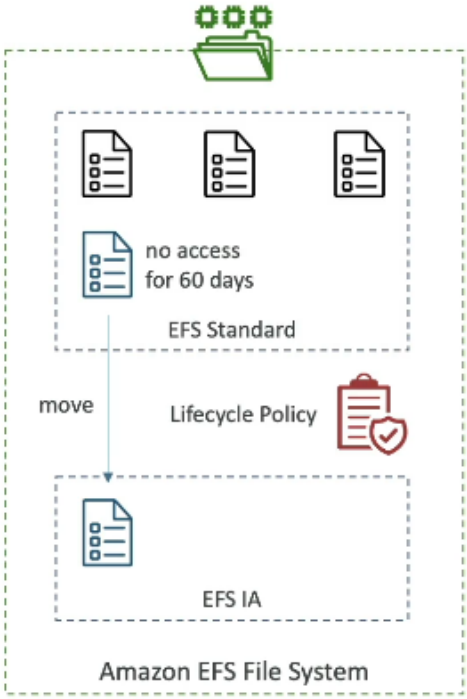
- **Performance Mode**
  - **General Purpose (default)** - best for latency-sensitive applications, such as web serving, content management systems, and development environments.
  - **Max I/O** - ideal for applications requiring high levels of aggregate throughput and parallelism, such as big data and 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

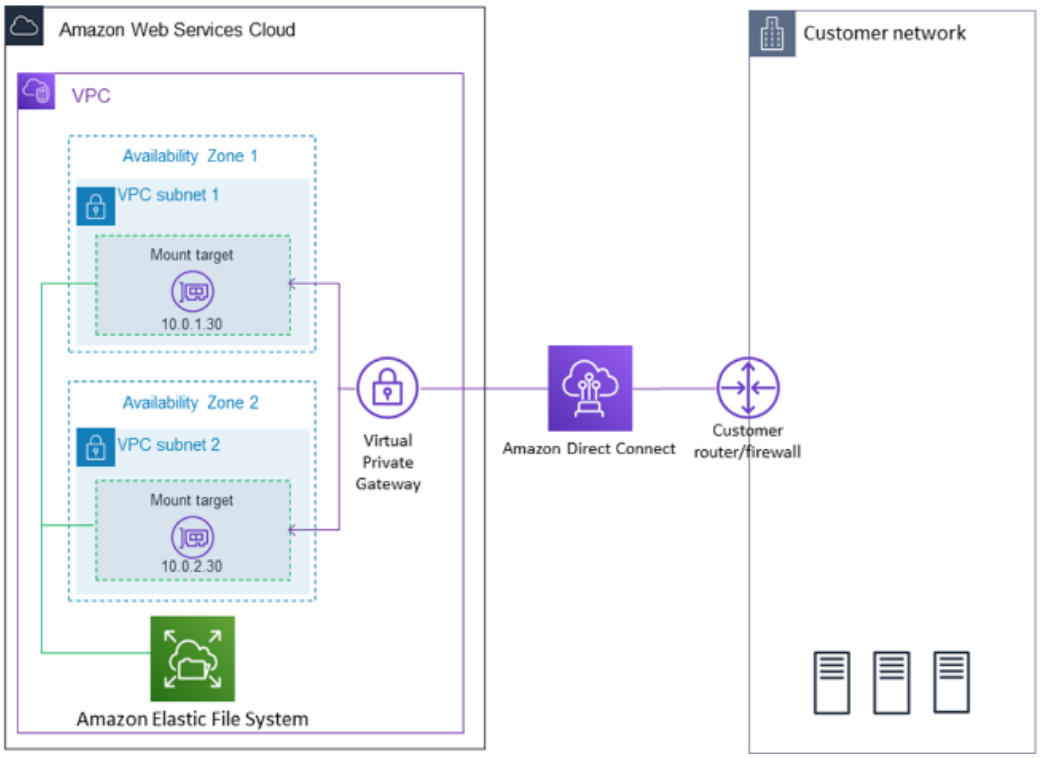
- **Standard** - for frequently accessed files. Data is redundantly stored across multiple Availability Zones within an AWS Region.
- **Infrequent Access (IA)** - for files that are accessed less frequently but require the same durability and availability as the EFS Standard class. It has lower price 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**).

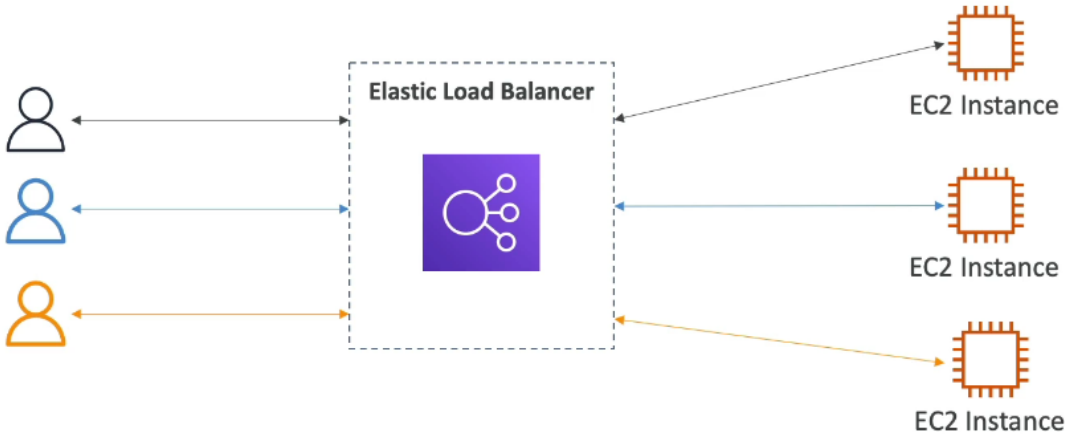
▼ ☑ **ELB.**

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.



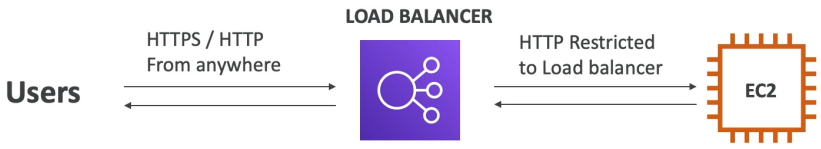
Health checks are crucial for Load Balancers. They enable the load balancer to know if instances it forwards traffict to are available to reply to requests. The health check is done on a port and a route (/health is common).

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

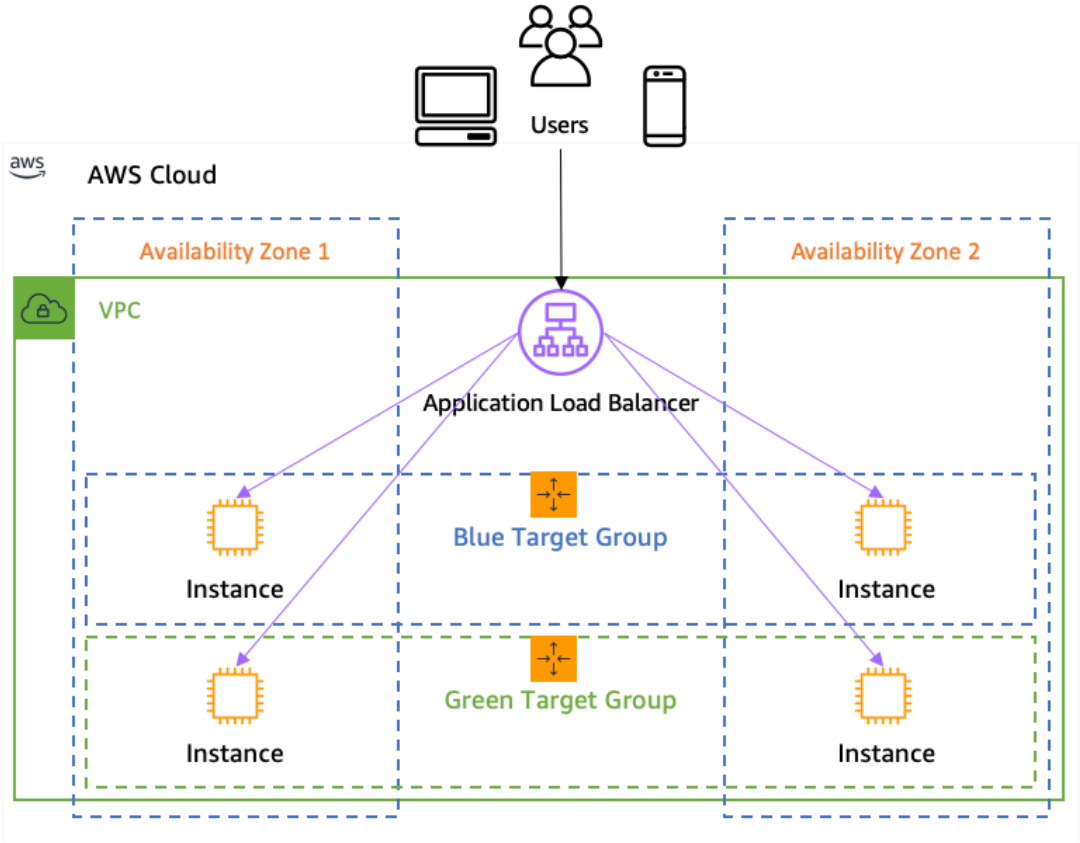
Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	sg-054b5ff5ea02f2b6e (load-b	Allow Traffic only...

▼ ☑ **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.

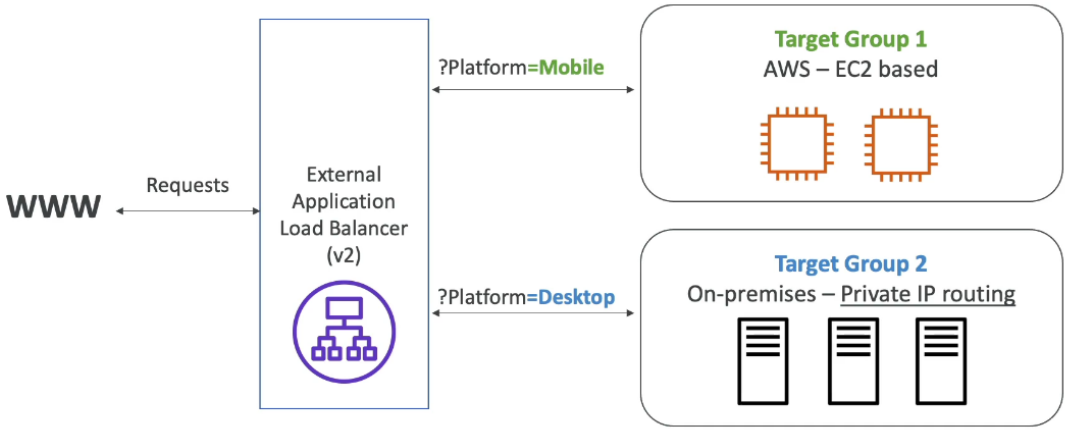


Target Groups:

- **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

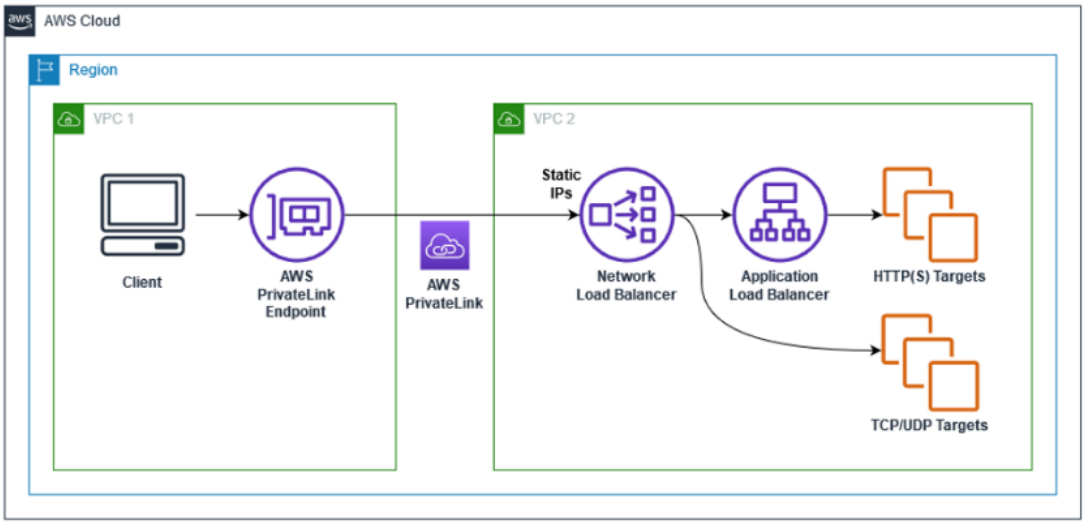
- **Path-based Routing** - based on the URL path (example.com/users).
- **Host-based Routing** - based on the Host field in the HTTP header (one.example.com).
- **Query String and Header-based Routing** - ased on the query string or specific headers in the HTTP request (exampe.com/users?id=123&order=false).

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).**

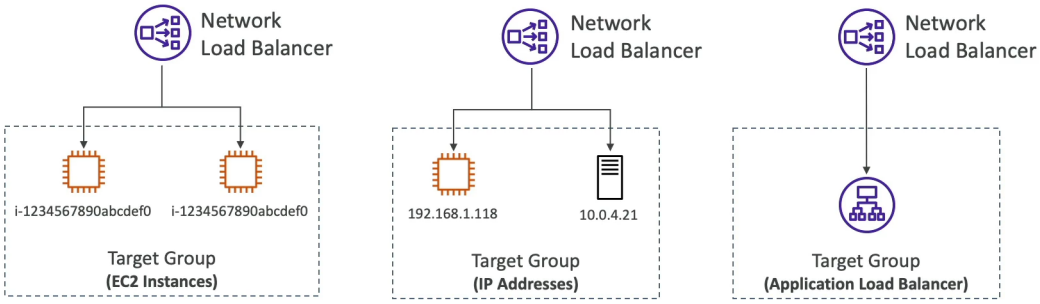
NLB operates at transport layer (Layer 4). Can handle high volumes of traffic with ultra-low latency, making it suitable for applications that require extreme performance, low latency, and TCP/UDP-level traffic handling. Network Load balancer doesn't have Weighted Target Groups.

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.

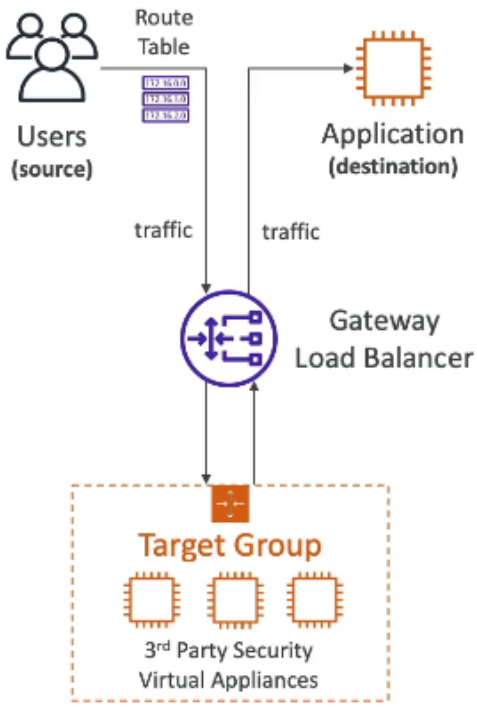
▼ ☑ **Gateway Load Balancer (GWLB).**

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:

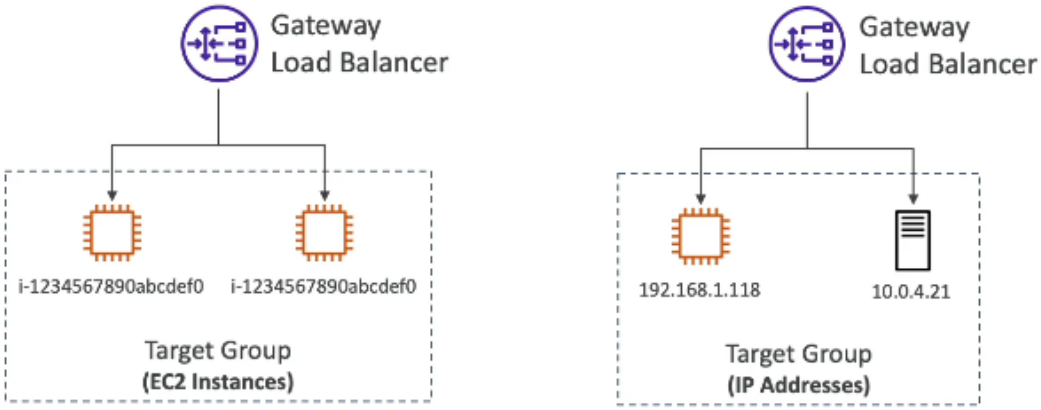
- Transparent Network Gateway - routes traffic without changing the source or destination IP addresses (**bump-in-the-wire**). It provides a single point of entry for all network traffic.
- Load Balancer - distributes traffic to our virtual appliances.

It uses the GENEVE protocol on port 6081.



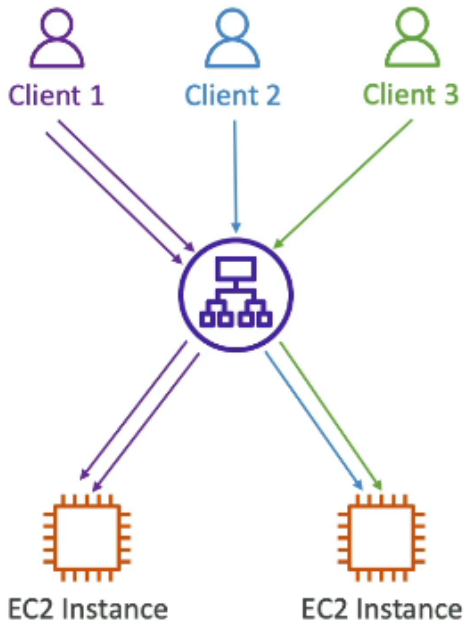
Target Groups:

- **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.

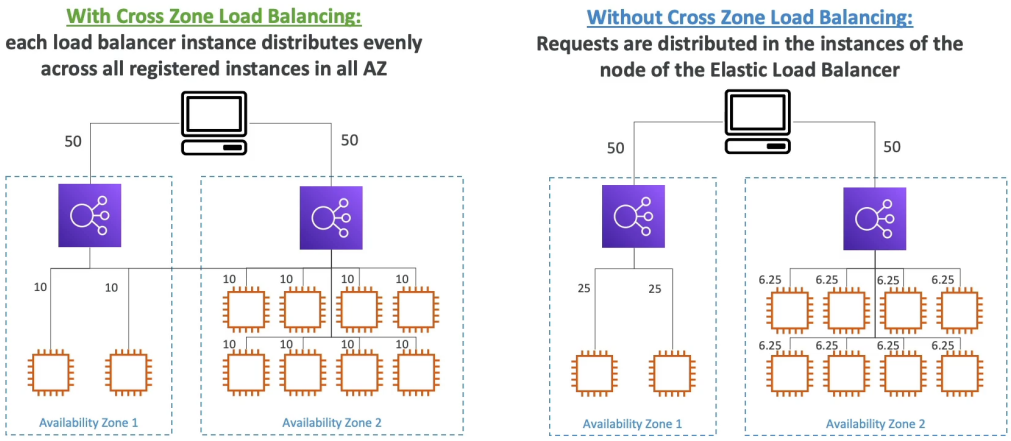
For ALB the cookie is used for stickiness. There are two types of cookies:

- **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).

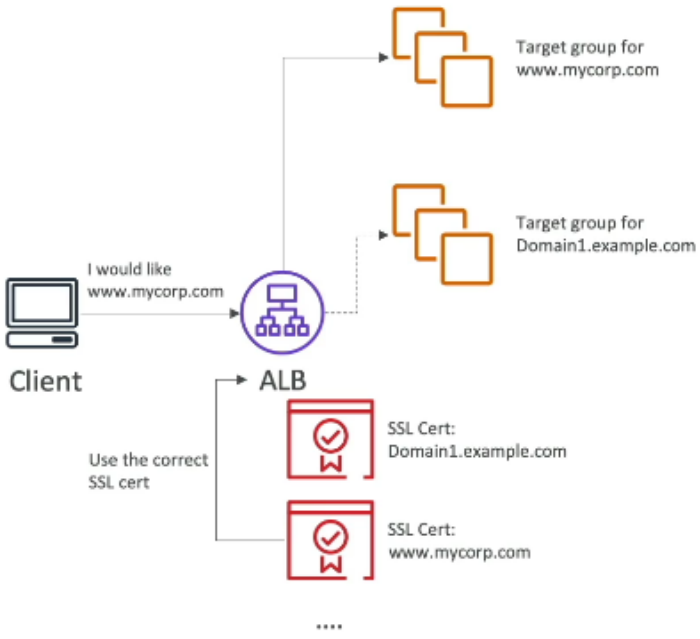
- SSL (Secure Sockets Layer) - used to encrypt connections.

- 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**.

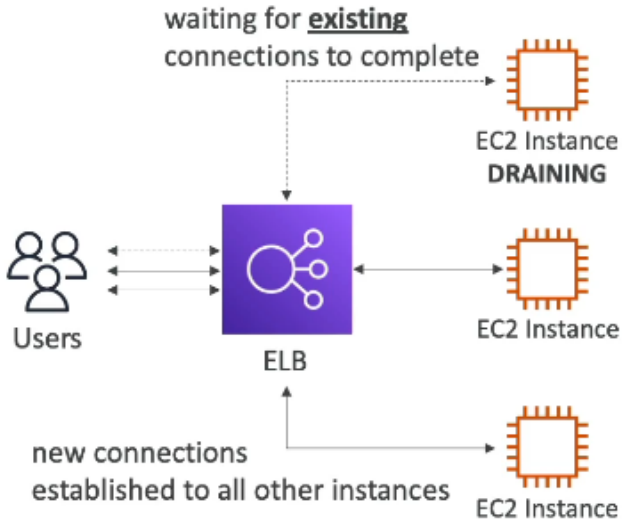


ALB & NLB support multiple listeners with multiple SSL certificates and use SNI to make it work.

▼ ☑ 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.

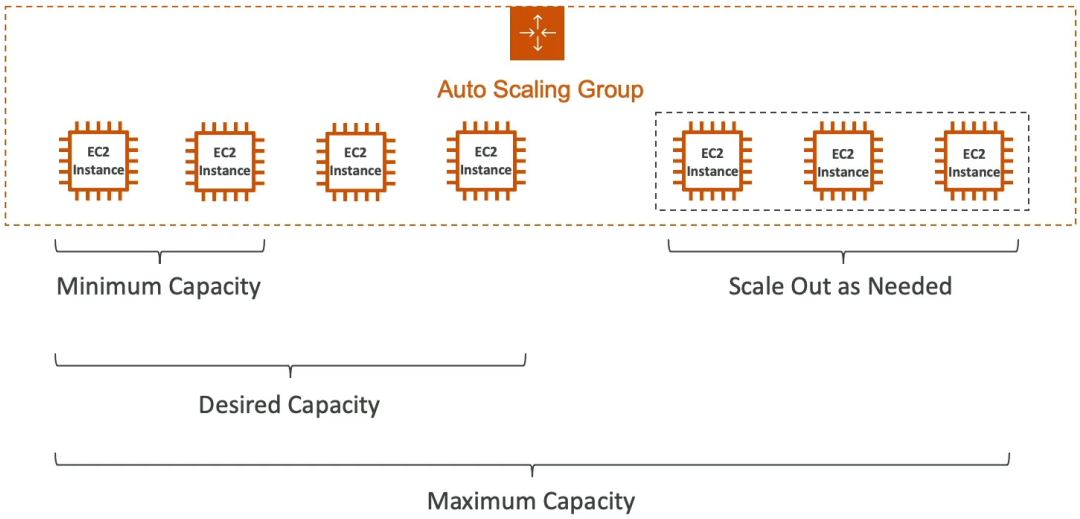
We can set timeout between 1 to 3600s (default is 300s). It can be disabled if we se value to 0. Good practice is to set to a low value if our requests are short.



▼ ☑ ASG.

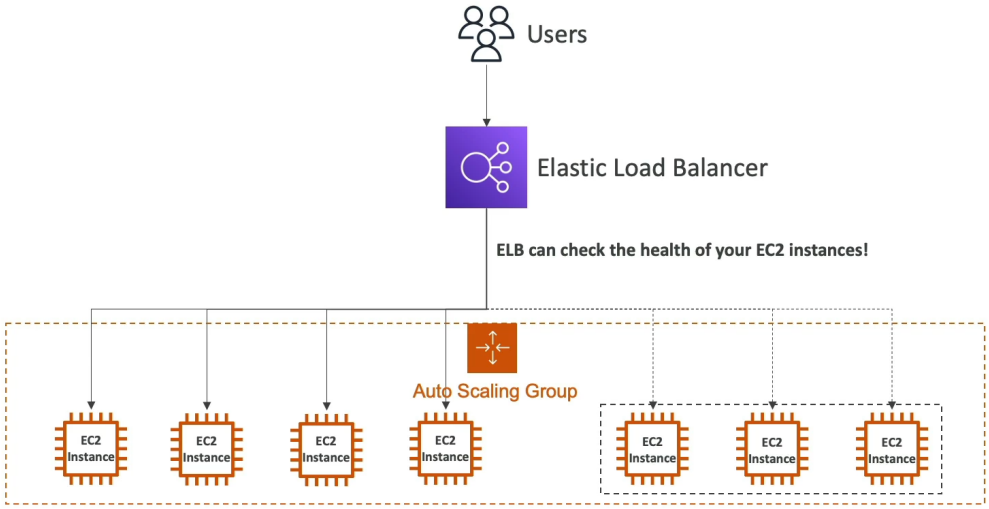
The goal of **ASG** is to:

- 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.



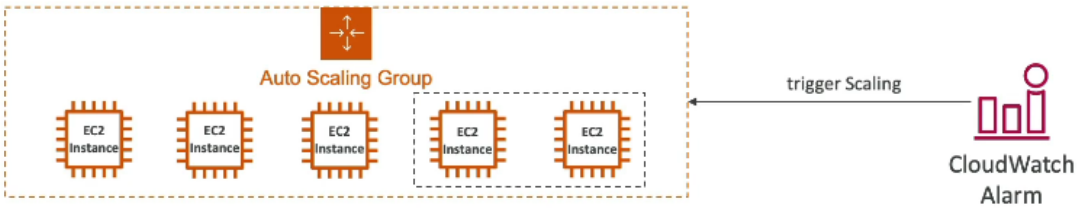
ASG Attributes

- **Launch Template**

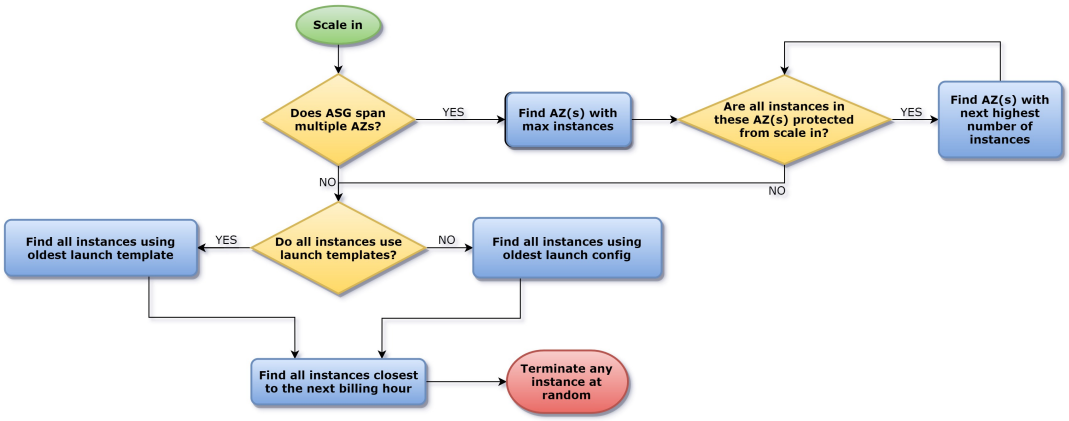
  It acts as a blueprint for EC2 instances, allowing us to define a set of configuration parameters that can be reused when we launch new instances. When instances are launched via a launch template, they can automatically be registered with an ELB. Once we have created a launch template, it can't be modified.

- **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.

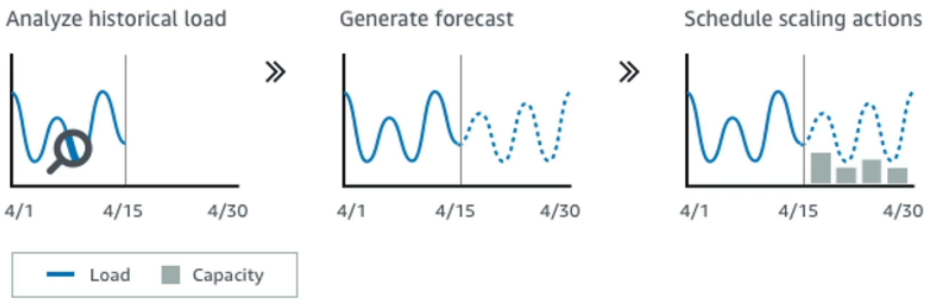
- **Target Tracking Scaling**

  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**

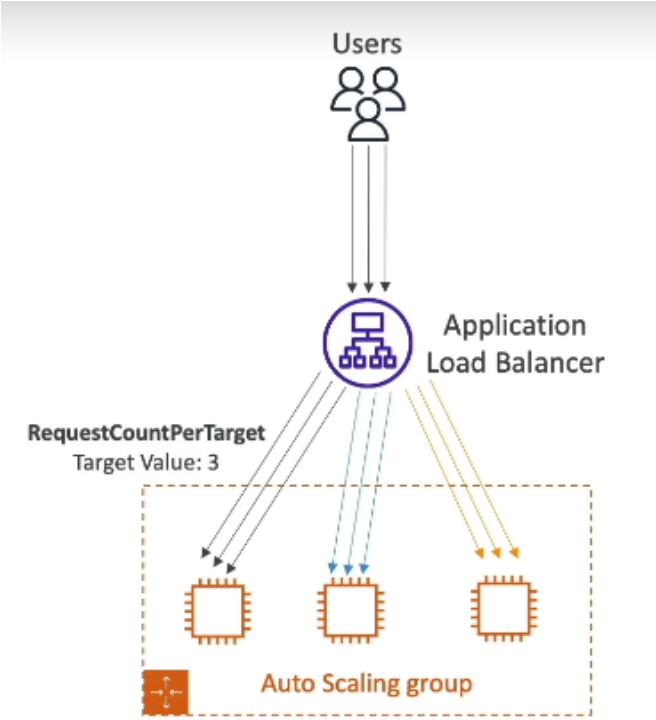
  Ideal for applications with predictable traffic patterns, such as regular business hours or batch processing jobs. Example: Increase the min capacity to 10 at 5PM on Fridays.

- **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.



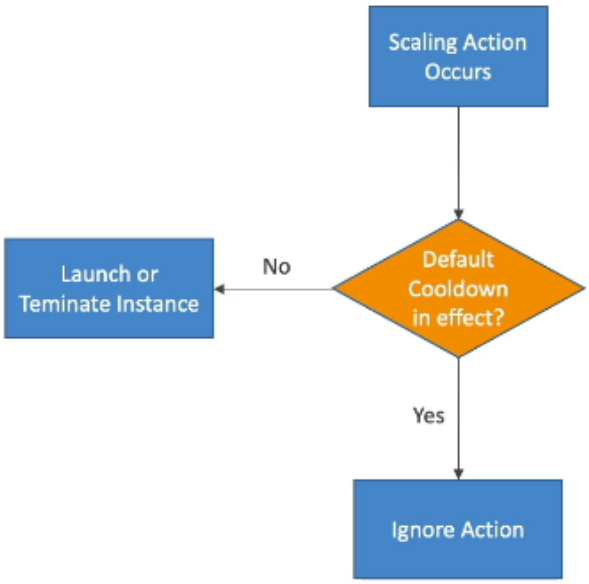
Good metrics to scale on:

- **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.

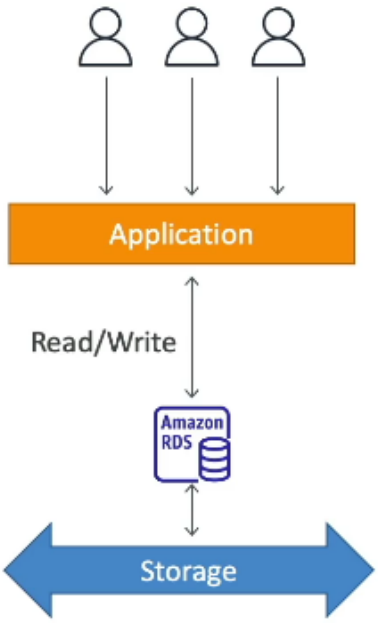
- Vertical and horizontal scaling capability.
- Storage backed by EBS.

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.



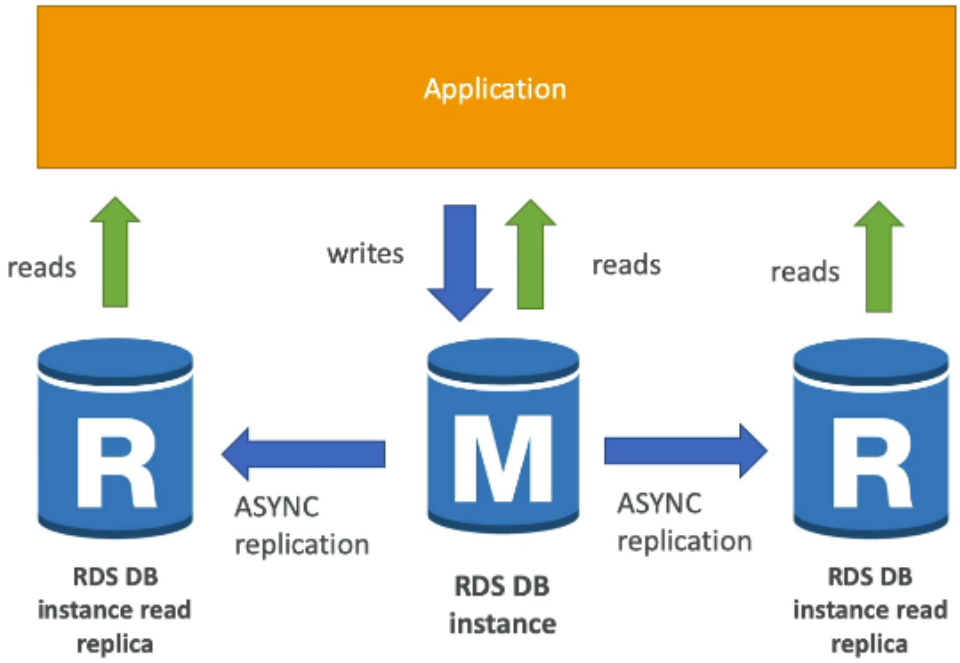
▼ ☑ **RDS Deployments.**

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

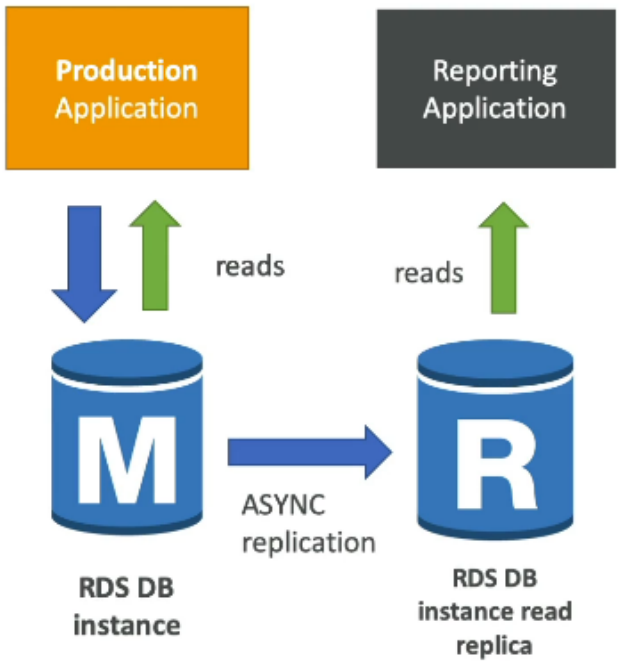
AZ or cross region. Applications must update the connection string to leverage read replicas.

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



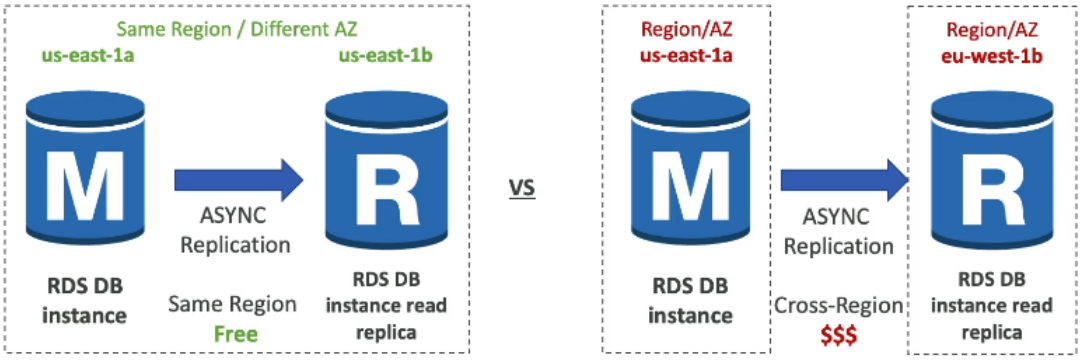
Read Replicas Use Cases:

- 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.

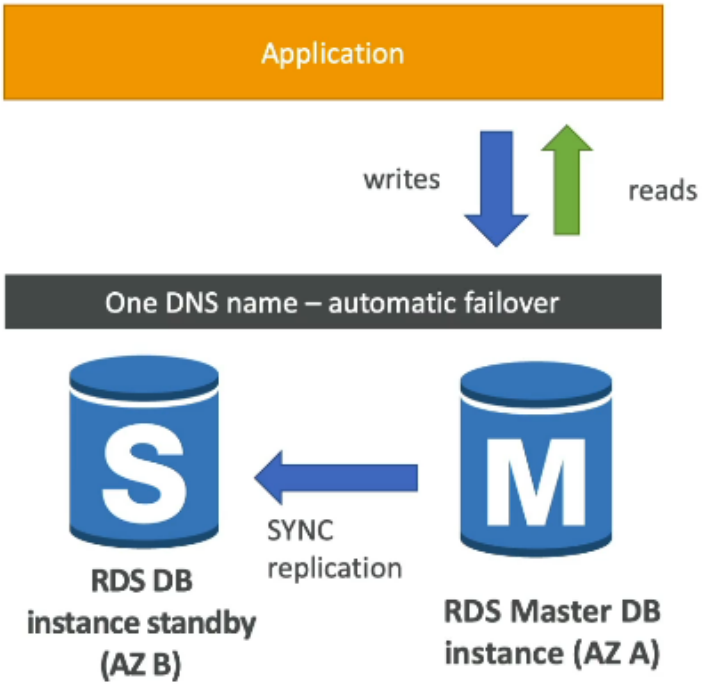
In AWS there is a network cost when data goes from one AZ to another. For RDS Read Replicas within the same region, we dont pay that fee.



Multi-AZ

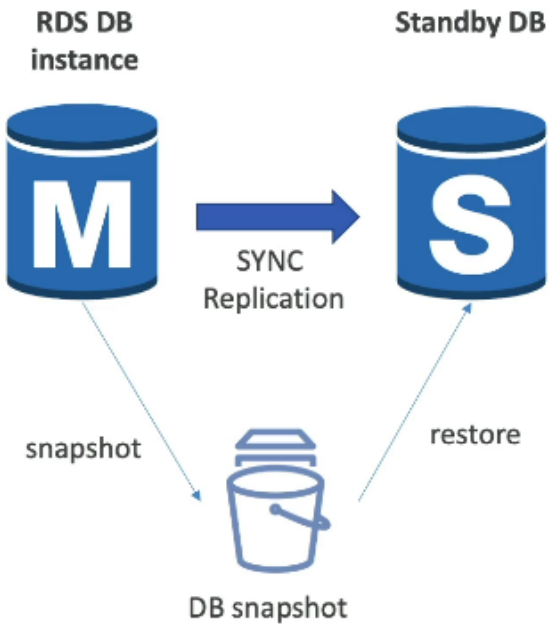
Multi-AZ deployments are designed to enhance the availability and durability of our database instances. This replication is **sync**. It is used for failover in case of loss of AZ, loss of network, isntace or storage failure. Not used for scaling.

> 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.

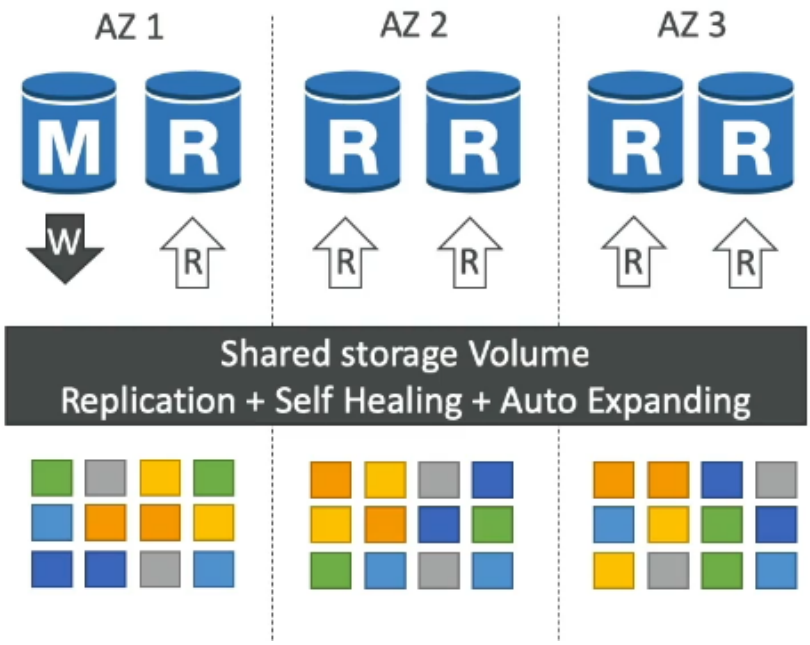
To perform our customization it is recommended deactivate Automation Mode (better to take a DB snapshot before).

▼ ☑ **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.

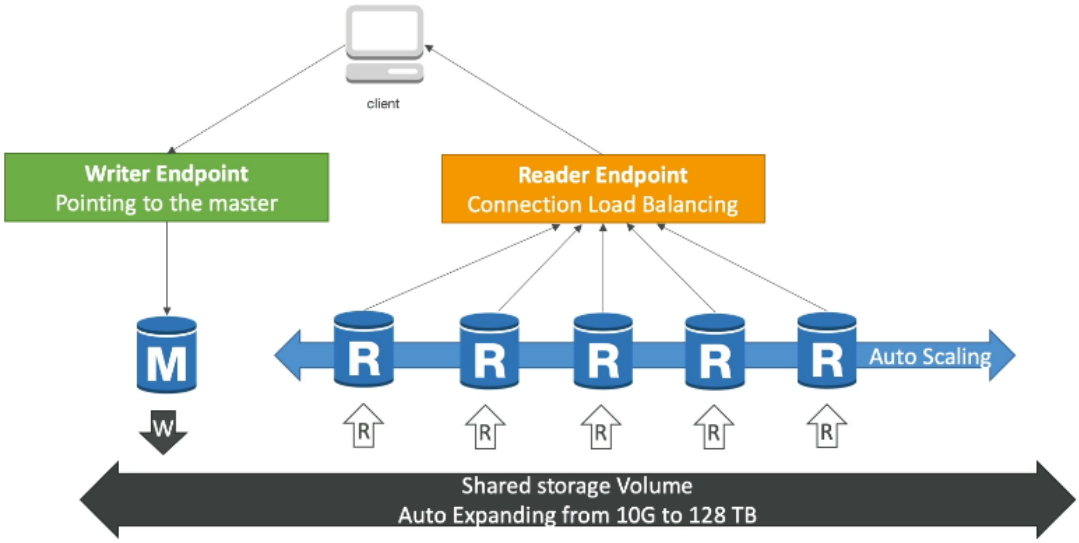


If we have an Aurora Replica in the same or a different AZ, when failing over, Aurora flips the canonical name record (CNAME) for our DB Instance to point at the healthy replica, which in turn is promoted to become the new primary. Start-to-finish failover typically completes within 30 seconds.

Aurora Endpoints

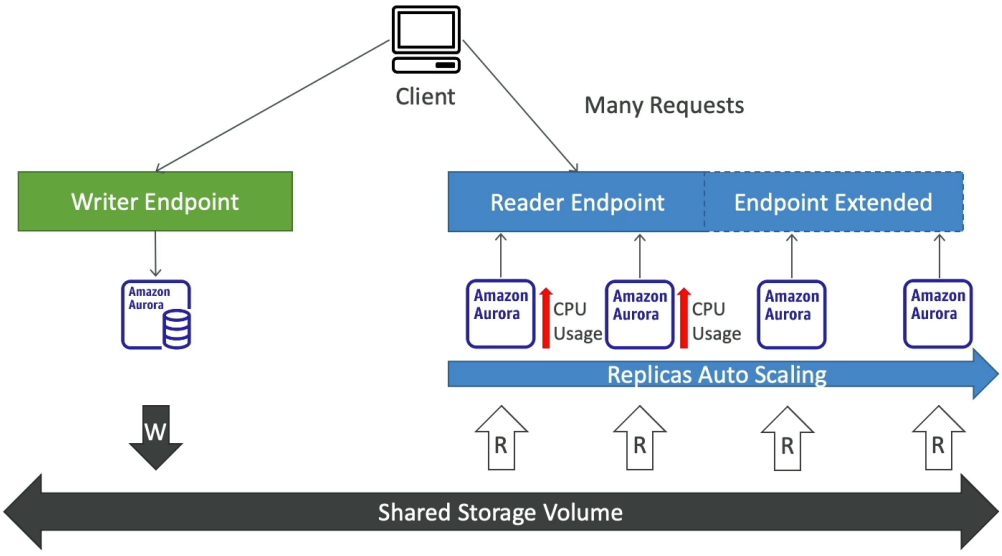
Writer Endpoint - used for all write operations to the Aurora database. This endpoint directs traffic to the primary instance within the Aurora cluster. In the event of a failure of the master instance, Aurora automatically promotes one of the replicas to be the new primary instance.

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.

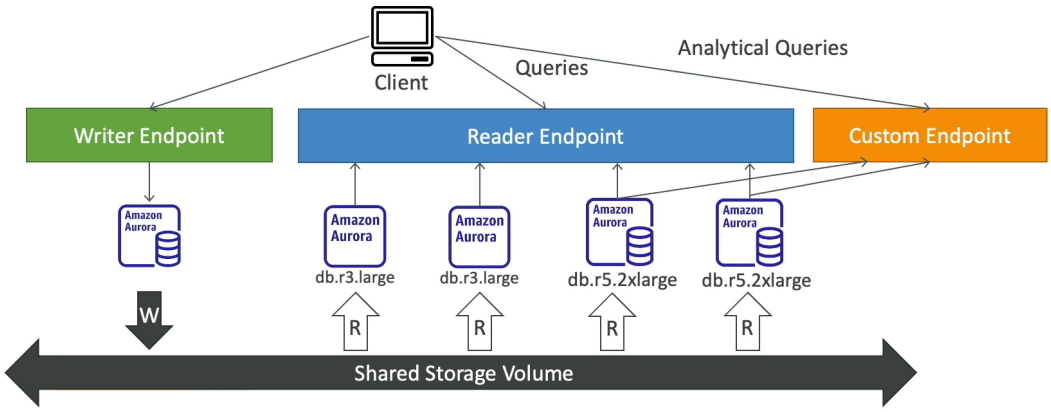


▼ ☑ **Aurora Advanced Concepts.**

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.

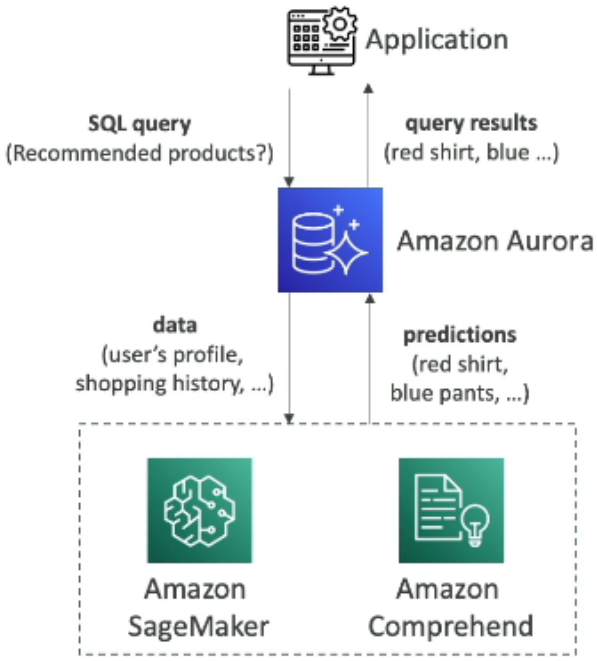


Aurora Serverless - automated database instantiation and auto-scaling based on actual usage (no management). No capacity planning is needed for it and we pay per second (can be more cost-effective). Can be used for infrequent or unpredictable workloads.

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.

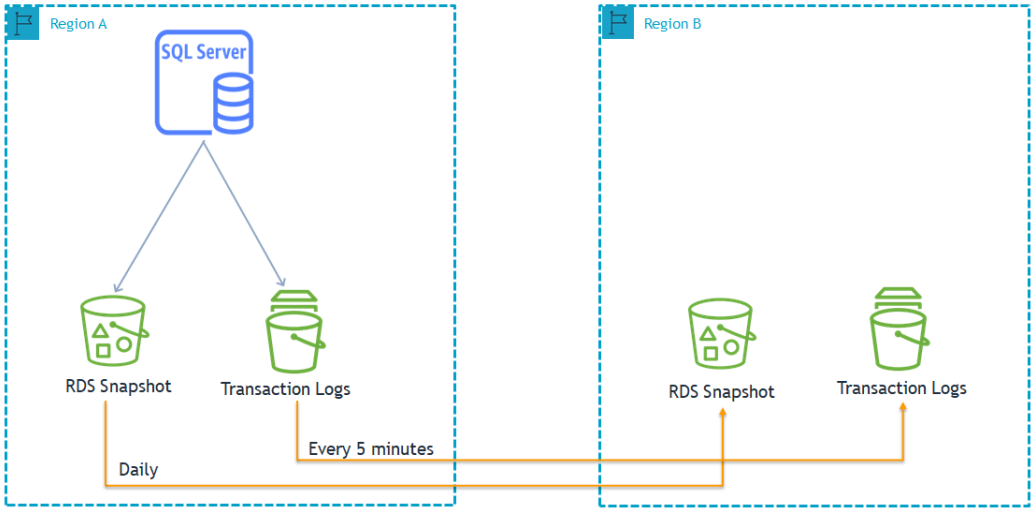
RDS Performance Insights is a feature that helps us monitor and analyze the performance of our RDS databases. It is offering a higher-level view of how our database queries and workloads are performing over time.

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.**

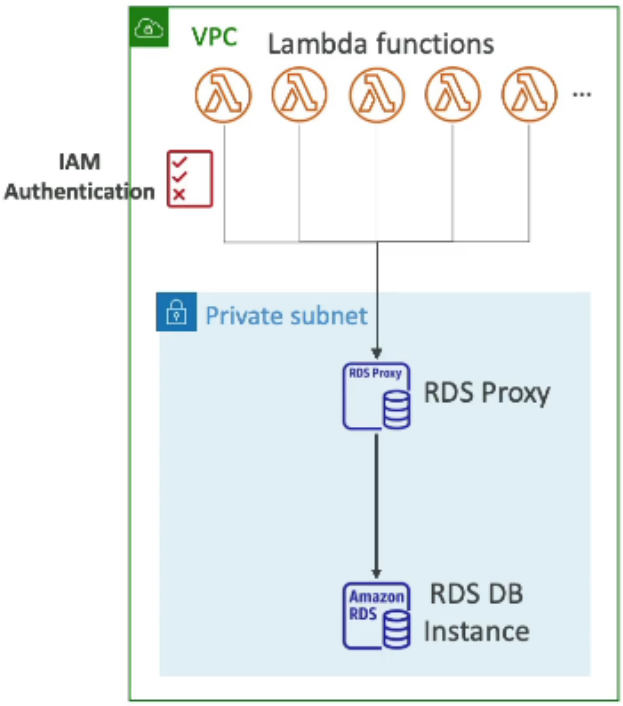
RDS Proxy is a fully managed (serverless, autoscaling, multi-AZ) database proxy for RDS. It allows apps to pool and share database connections established with

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

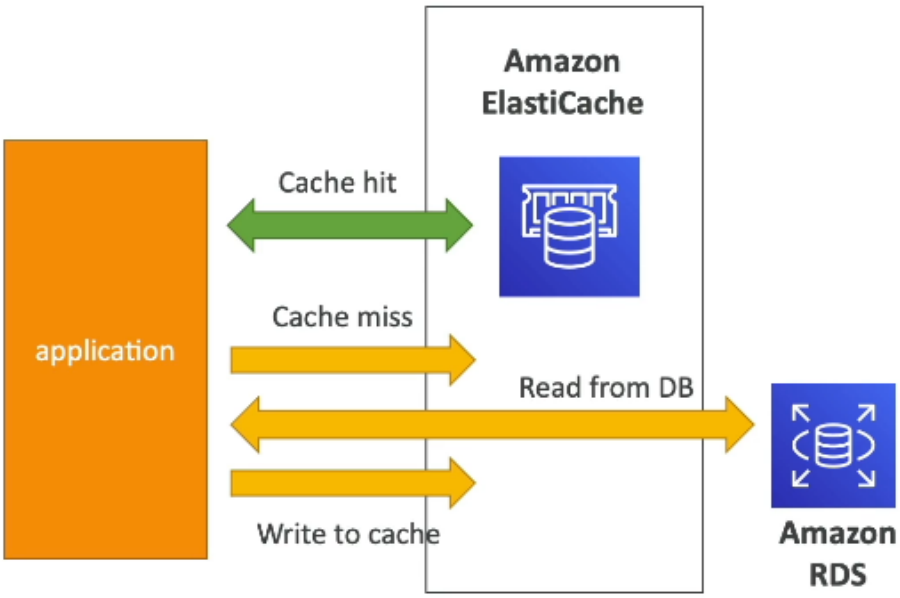
REDIS	MEMCACHED
• Multi AZ with Auto-Failover • Read Replicas to scale reads and have high availability • Data Durability using AOF persistence • Backup and restore features • Supports Sets and Sorted Sets	• Multi-node for partitioning of data (sharding) • No high availability (replication) • Non persistent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



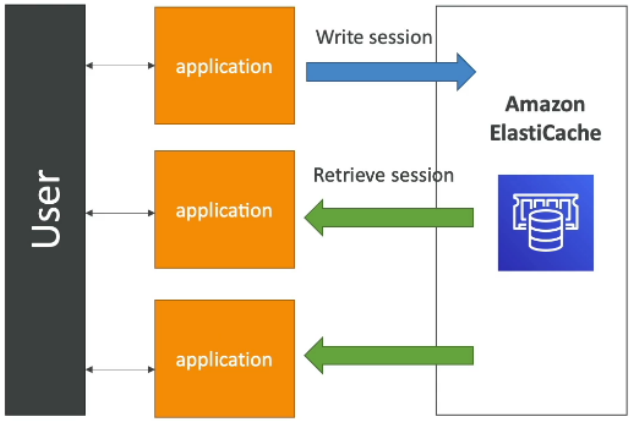
Cache must have an invalidation strategy to make sure only the most current data is used in there.

▼ **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:

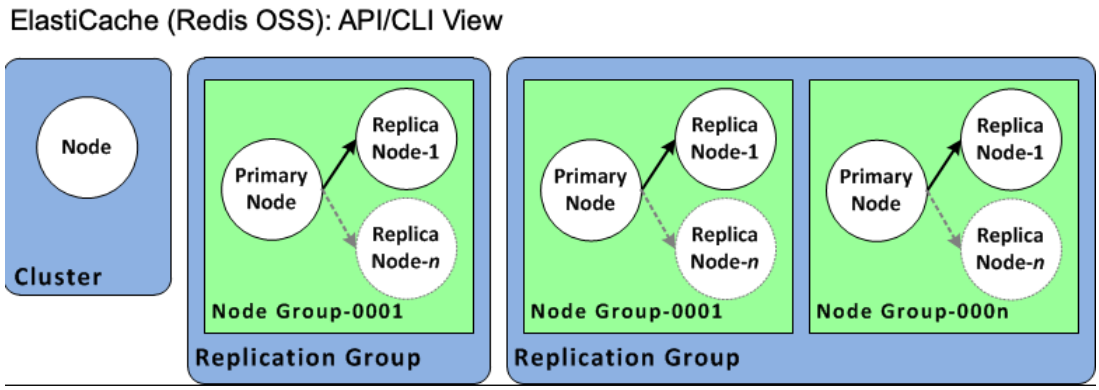
- **Amazon ElastiCache (Redis)**
- **Application Servers**
- **User Database (Optional)**



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.

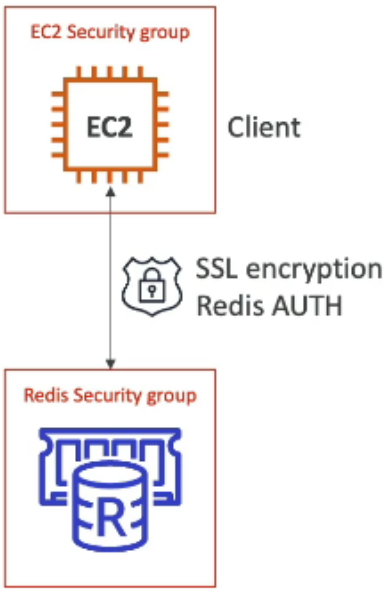
▼ ☑ **ElastiCache Advanced Concepts.**

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.

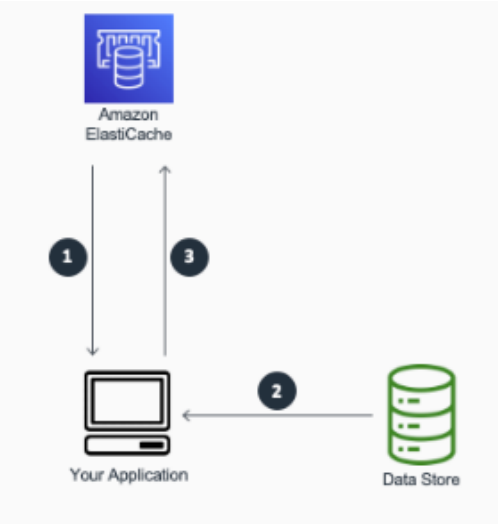
To require that users enter a password on a password-protected Redis server, include the parameter --auth-token with the correct password when we create our replication group or cluster and on all subsequent commands to the replication group or cluster.



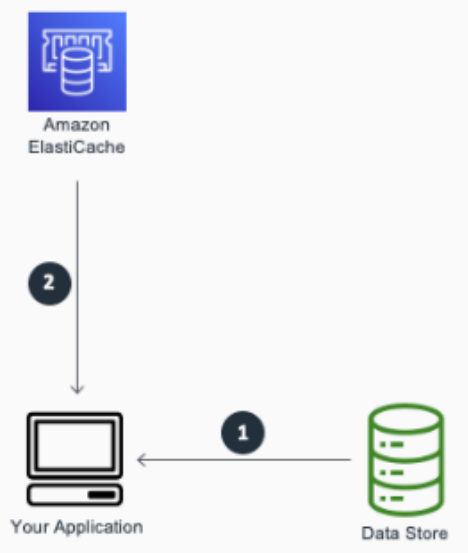
Memcached supports SASL-based authentication.

ElastiCache Design Patterns

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



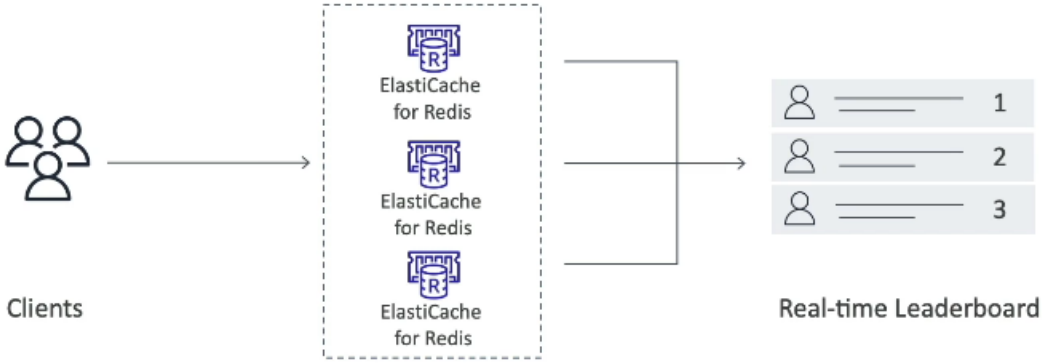
- **Write Through** - adds or updates data in the cache written to a database (no stale data).



- **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.

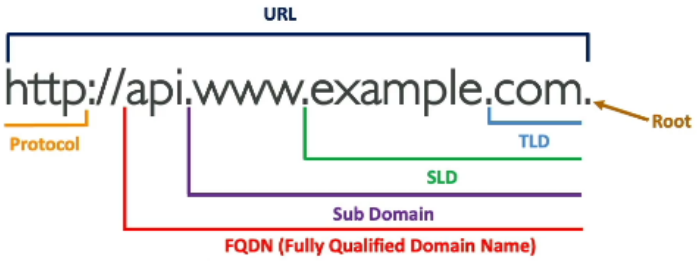


Route 53

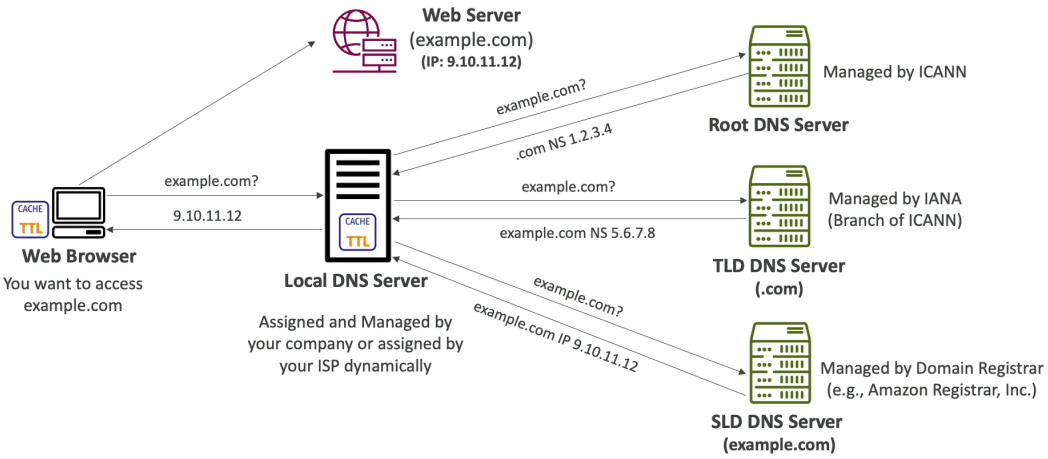
▼ ☑ DNS.

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.

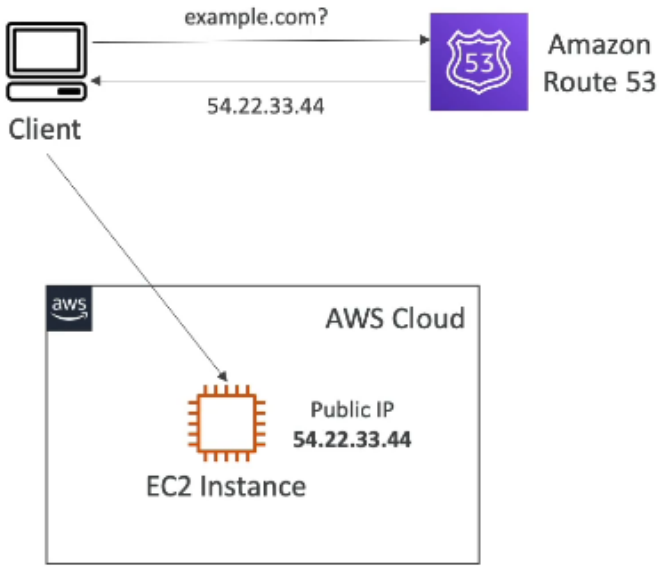


How DNS Works



▼ ☑ **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:

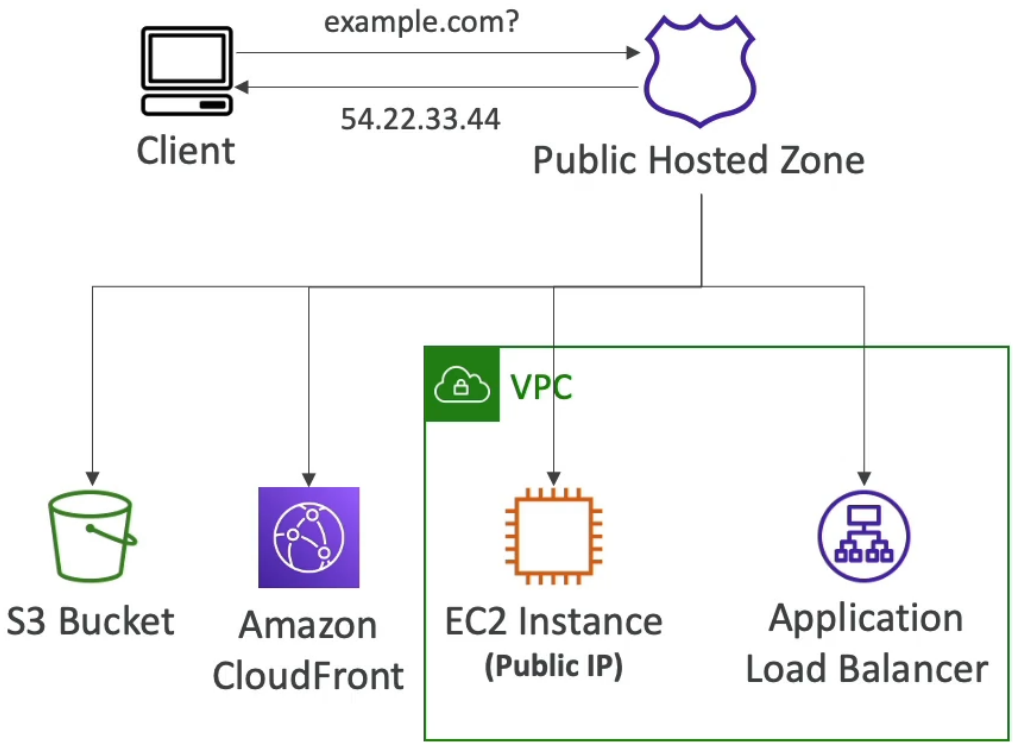
- **Domain/subdomain Name** (e.g. example.com)
- **Record Type** (e.g. A, AAAA)
- **Value** (e.g. 12.34.56.78)
- **Routing Policy** - how Route 53 responds to queries.
- **TTL (Time to Live)** - amount of time the record is cached at DNS Resolvers.

Record Types

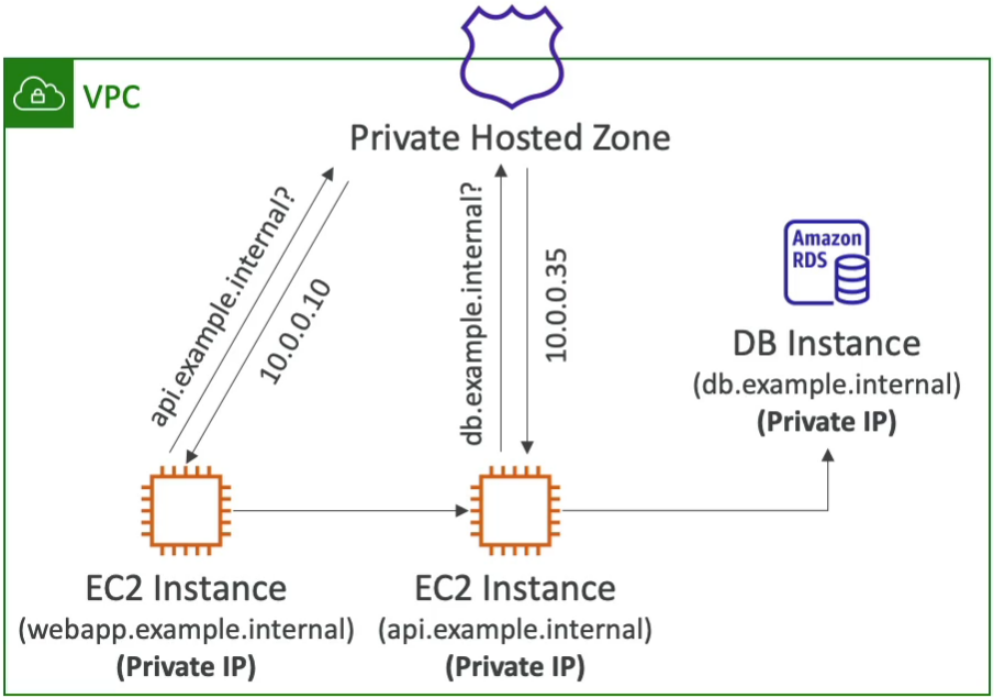
- A - maps a domain name to an IPv4 address.
- AAAA - maps a domain name to an IPv6 address.
- CNAME - maps hostname to another hostname. The target is a domain name which must have an A or AAAA record. We cant create a CNAME record for the top node of DNS namespace (Zone Apex).
- NS - specifies the 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.

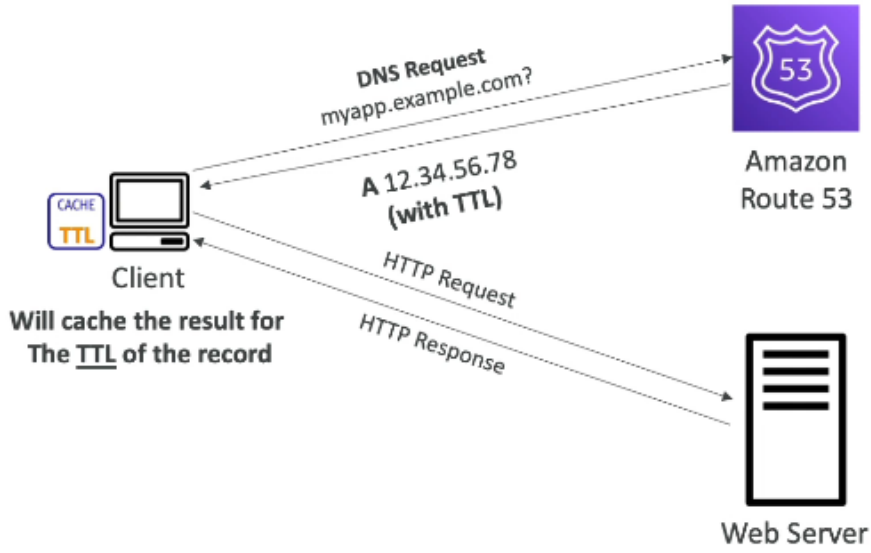
▼ ☑ **Route 53 TTL.**

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.



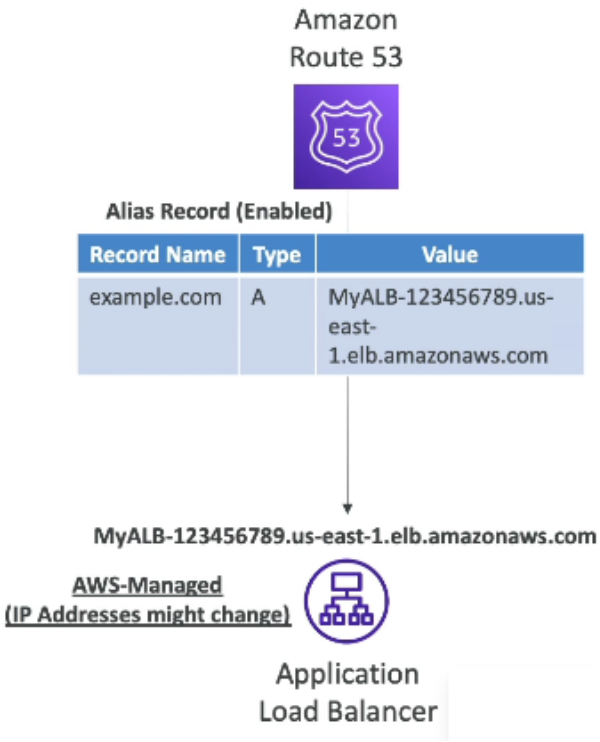
▼ ☑ **CNAME & Alias Records.**

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

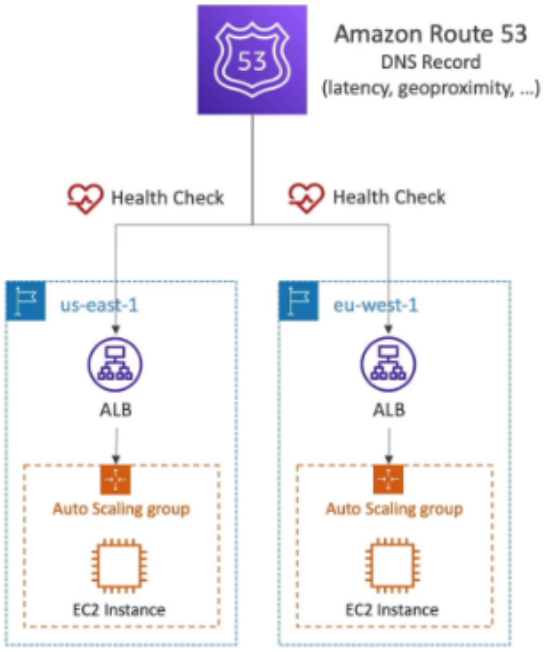
- Elastic Load Balancers
- CloudFront Distributions
- API Gateway
- Elastic Beanstalk environments
- S3 Websites
- VPC Interface Endpoints
- Global Accelerator accelerator
- Route 53 record in the same hosted zone

- 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.

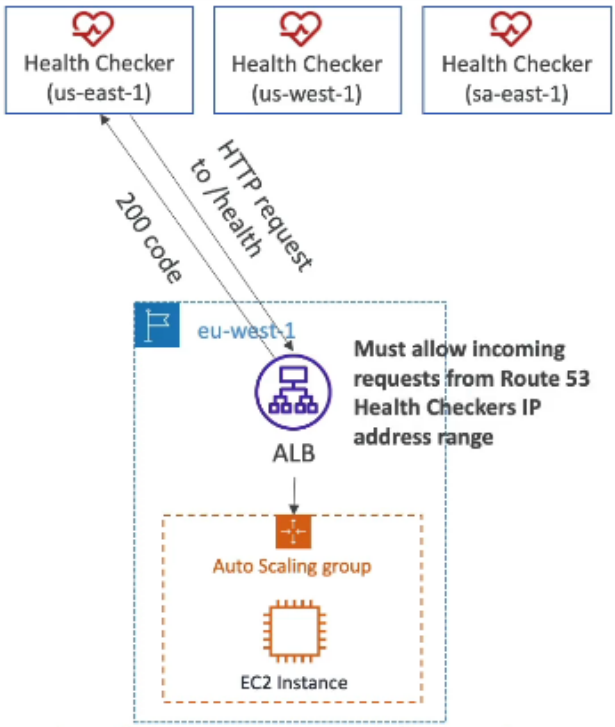


- Health Checks that monitor an enpoint

  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.

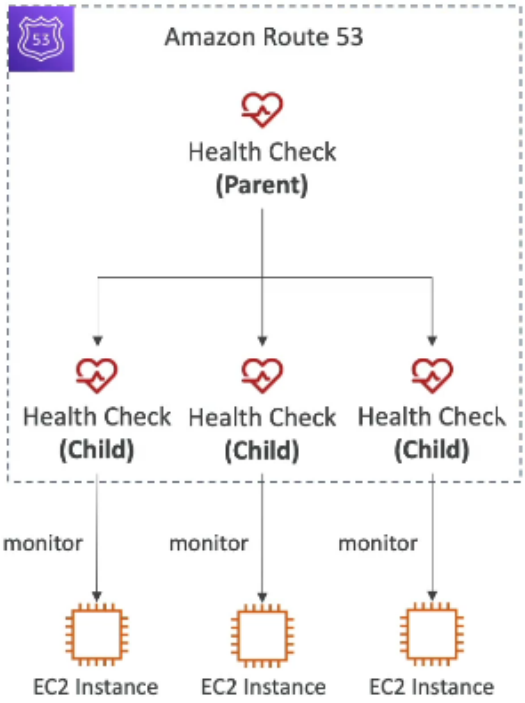
  To make it work we must configure our router/firewall to allow incoming requests from Route 53 Healtch Checkers.



- Calculated Health Checks

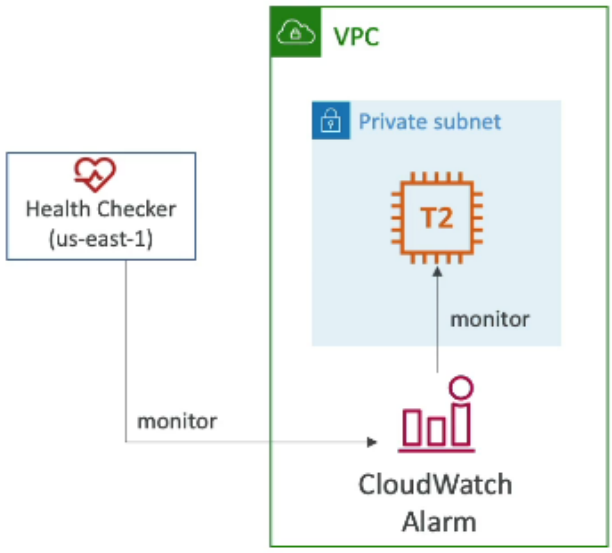
  It combines the reuslts of multiple health checks into a single health check. We can use OR, AND or NOT operators and can specify how many of the health checks need to pass to make the parent pass.

  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).

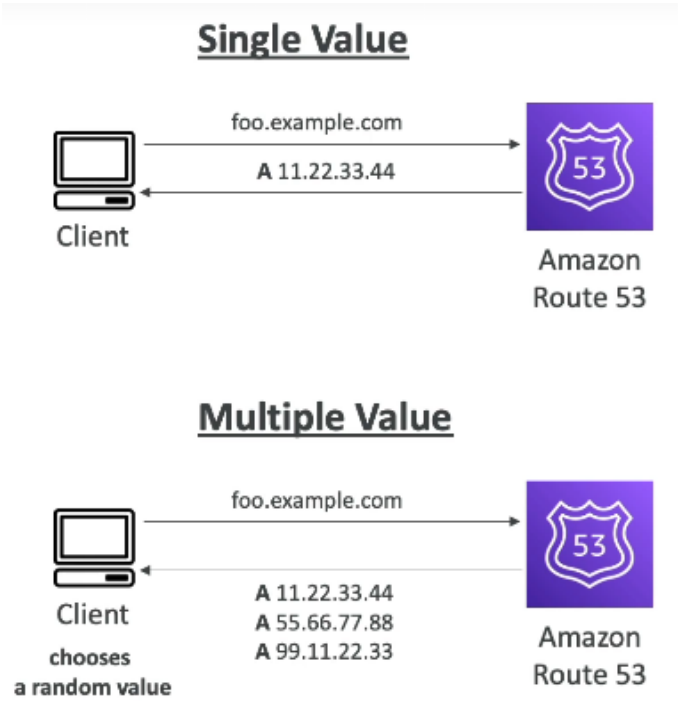
  Because health checkers are outside the VPC, they cant access private endpoints. We can create CloudWatch Metric and associate a CloudWatch Alarm, then create a health check that checks the alarm itself.



▼ ☑ **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.



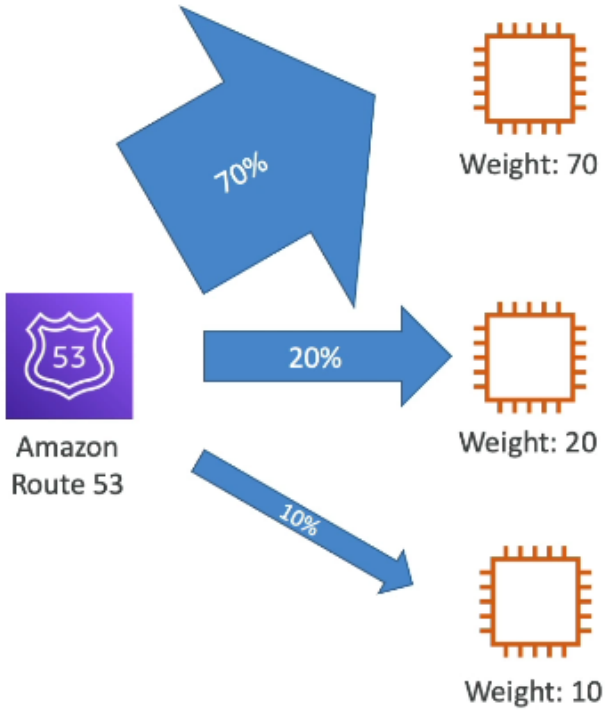
▼ ☑ **Routing Policy - Weighted Routing.**

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.**

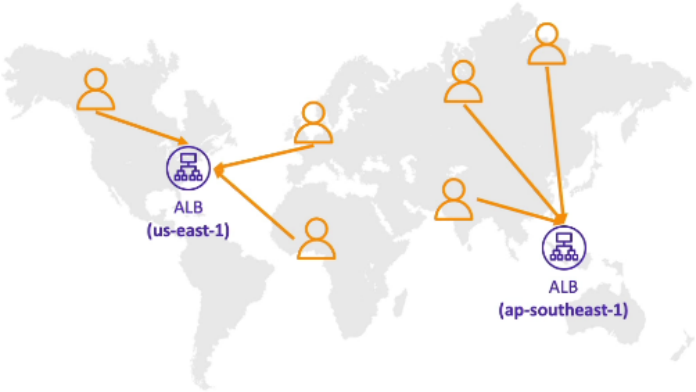


Use cases: load balancing between regions, testing new application versions ...

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

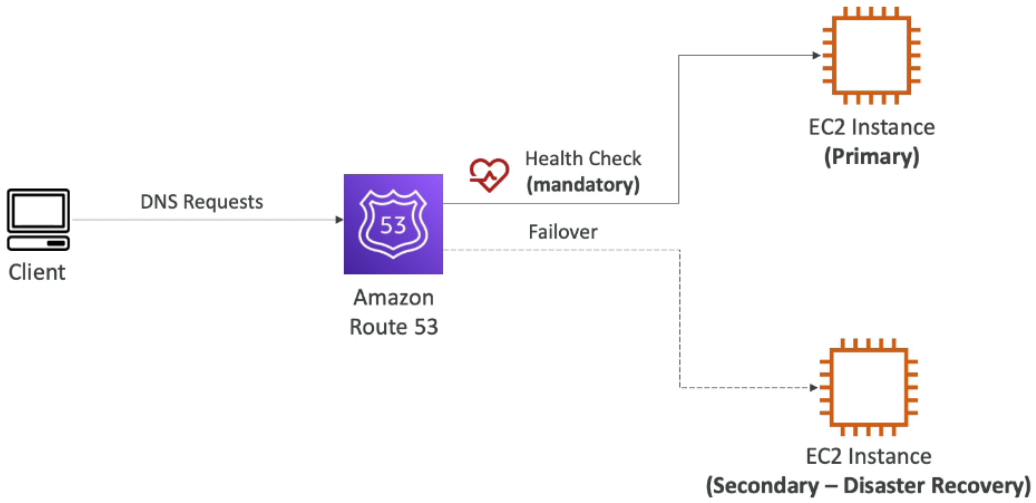
It redirects to the resource that has the least latency close to us. Latency is based on traffic between users and AWS regions.

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).

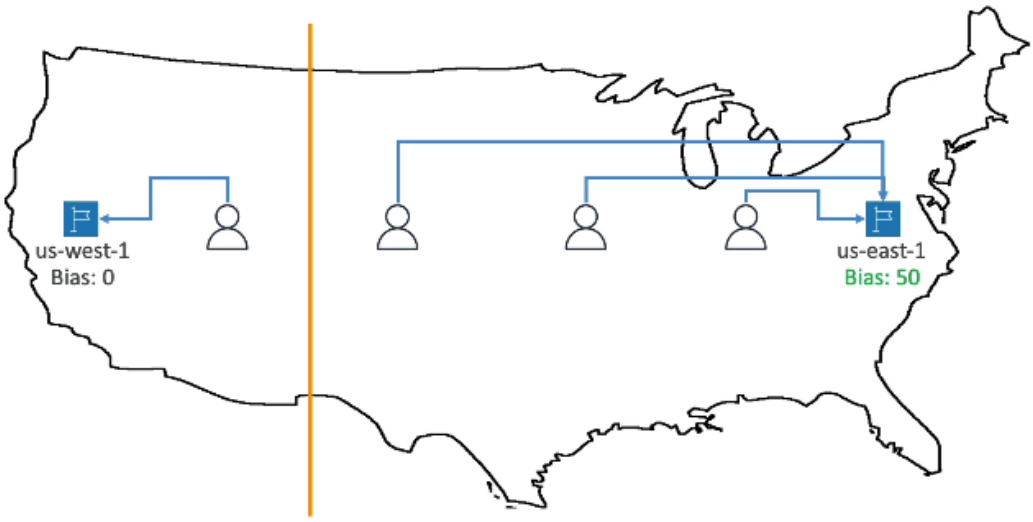
We should create a “Default” record (in case there is no match on location). Also, it can be associated with Health Checks.

▼ ☑ **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

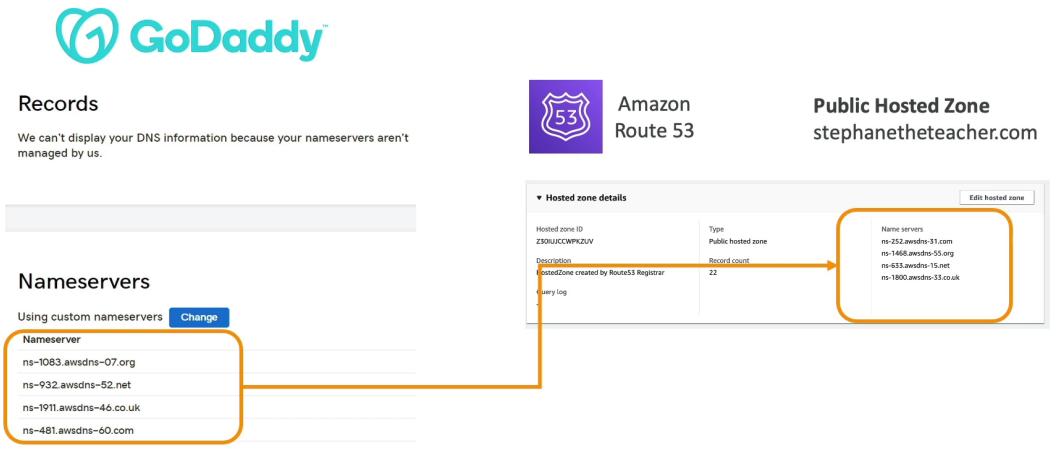
Multi-Value is not a substitute for having an ELB.

▼ ☑ **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

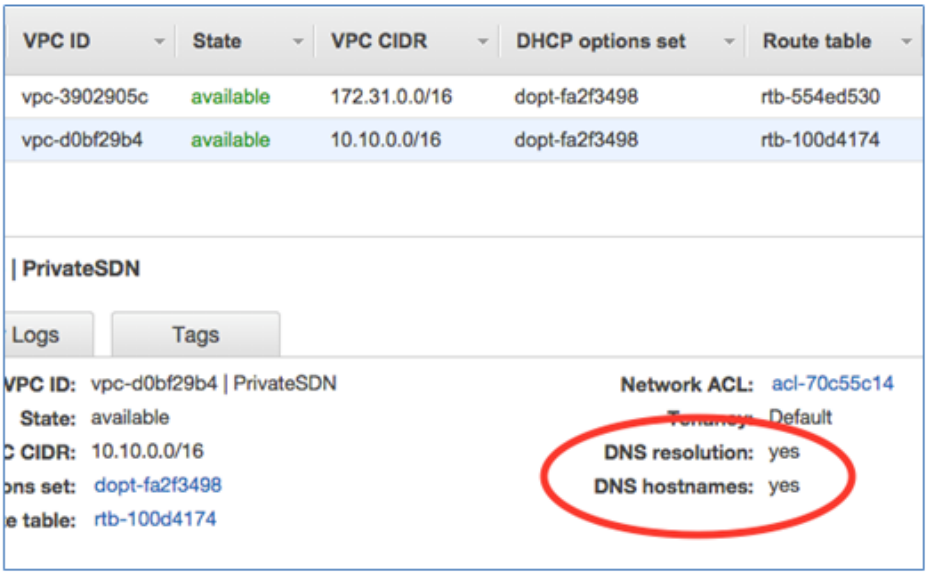
manage our DNS records, but we can use another DNS service to manage our DNS records.

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

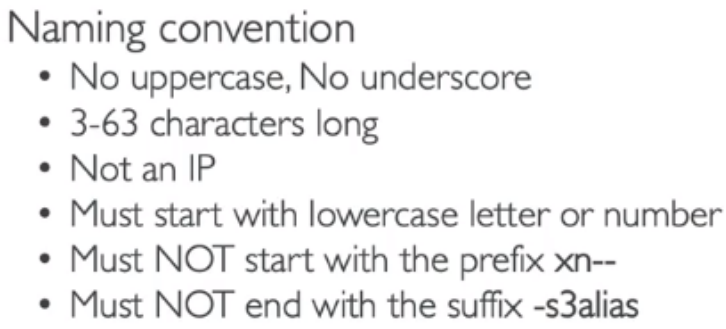
Amazon S3

▼ ☑ **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.

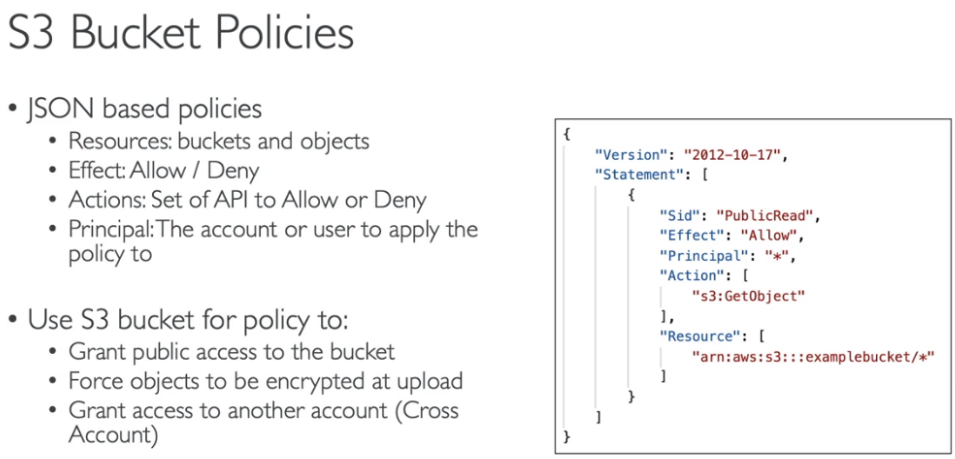
Object values are content of the body. Max object size is 5TB, if we upload more than 5GB, must use "multi-part upload". Object can have also **metadata** (list of text key/value pairs), **tags** (unicode key/value pair, useful for security/lifecycle), **version ID**.

▼ ☑ **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.

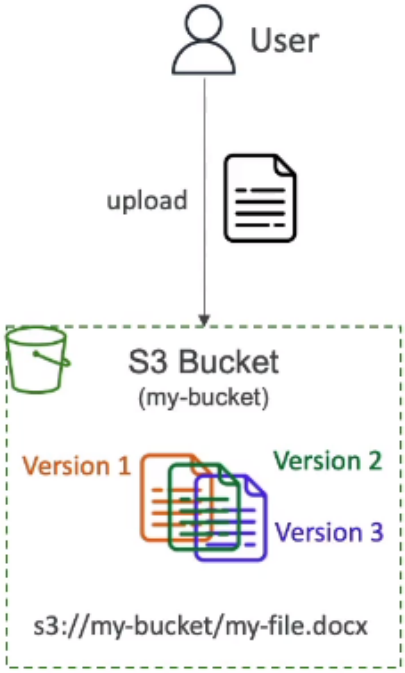
By default, an S3 object is owned by the AWS account that uploaded it even though the bucket is owned by another account. Object owner must explicitly grant the bucket owner access using a bucket policy to require external users to grant *bucket-owner-full-control* when uploading objects.

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.



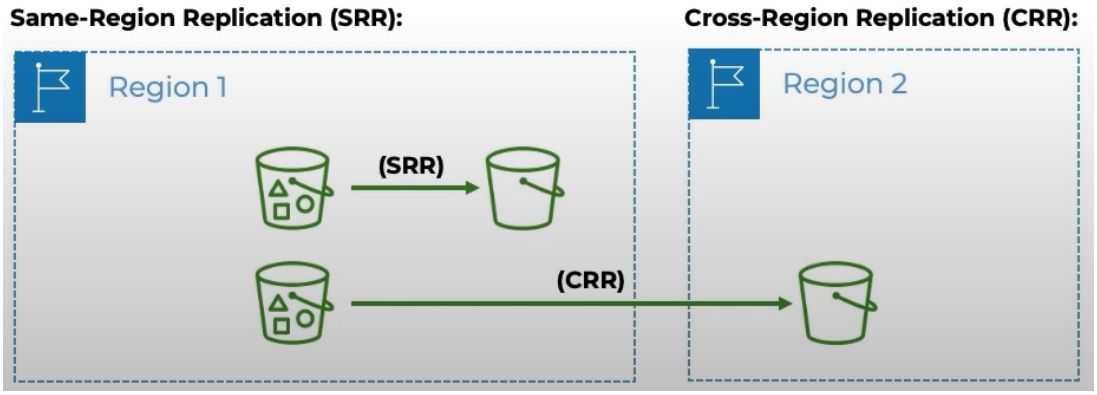
It's best practice to version our buckets. It protects us against unintended deletes (can restore a version) and we can roll back to previous version easily.

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.

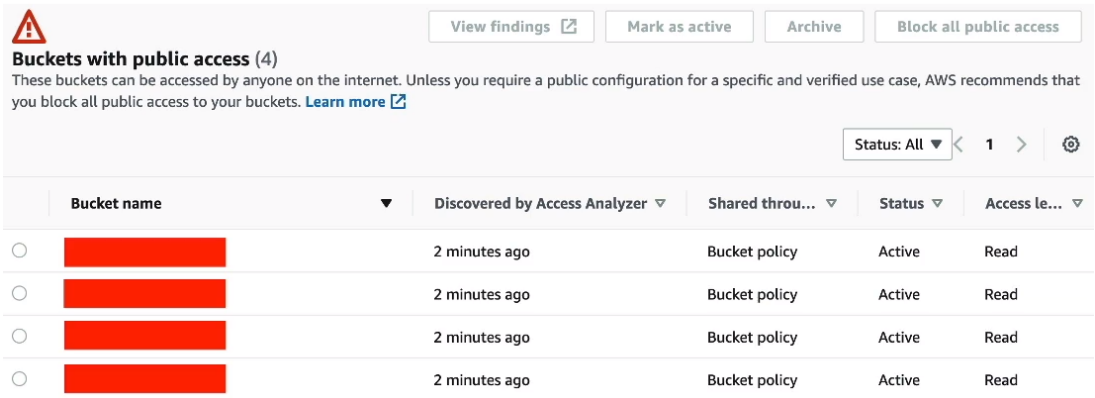
	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Durability	99.999999999% == (11 9's)						
Availability	99.99%	99.9%	99.9%	99.5%	99.9%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99%	99.9%	99.9%
Availability Zones	>= 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.**

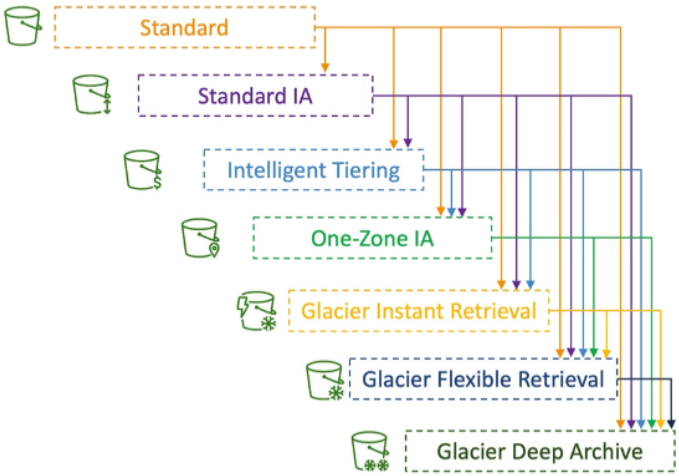
IAM Access Analyzer ensures that only intended people have access to our S3 buckets (for example: publicly accessible bucket or bucket shared with other AWS account).

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

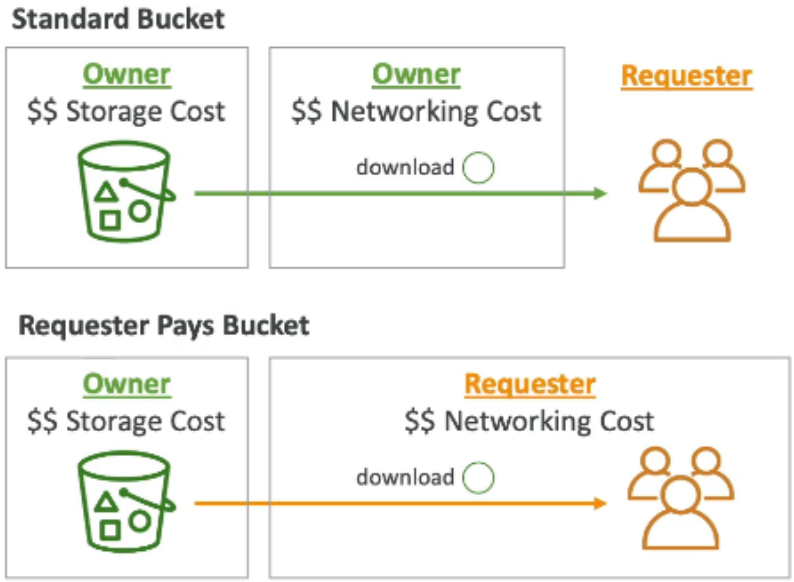
this time, and for up to 365 days, deleted objects should be recoverable within 48 hours.

- 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.**

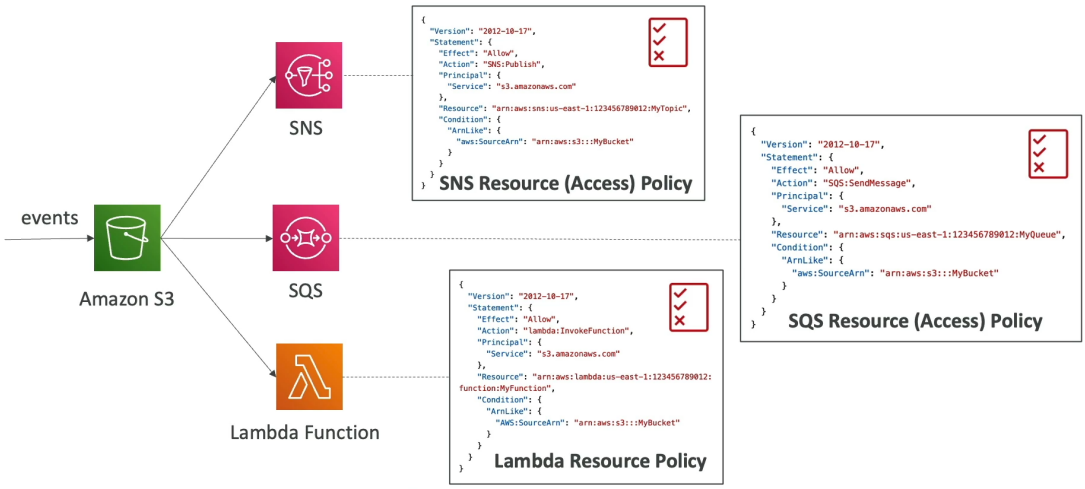
In general, buckets owners pay for all S3 storage and data transfer costs associated with their buckets. With Requester Pays buckets, instead of the bucket owner, the requester pays the cost of the request and the data download from the bucket.



The requester must be authenticated in AWS.

▼ ☑ **S3 Event Notifications.**

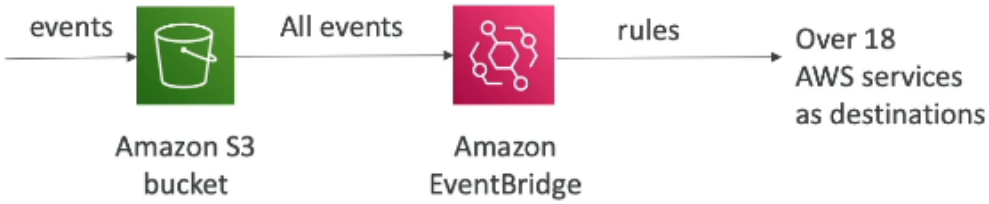
S3 Event Notifications allow us to receive notifications when certain events happen to objects within our S3 buckets. We can create as many S3 events as desired.



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.

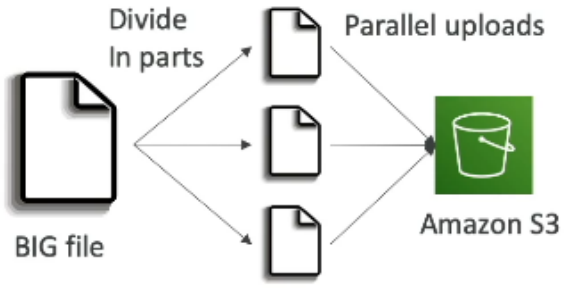


▼ ☑ S3 Performance.

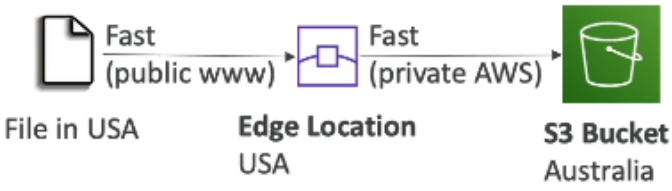
S3 automatically scales to high request rates, latency 100-200ms. Our application can achieve at least 3500 PUT/COPY/POST/DELETE or 5500 GET/HEAD requests per second per prefix in a bucket. If we spread reads across all four prefixes evenly, we can achieve 22000 requests per second for GET and HEAD.

Performance Boosts

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



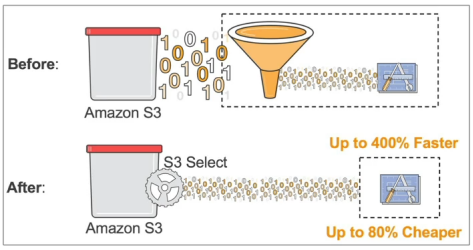
- **S3 Transfer Acceleration** - increase transfer speed by transferring file to an AWS Edge Location which will forward the data to the S3 bucket in the target region. It is compatible with multi-part upload.



- **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.

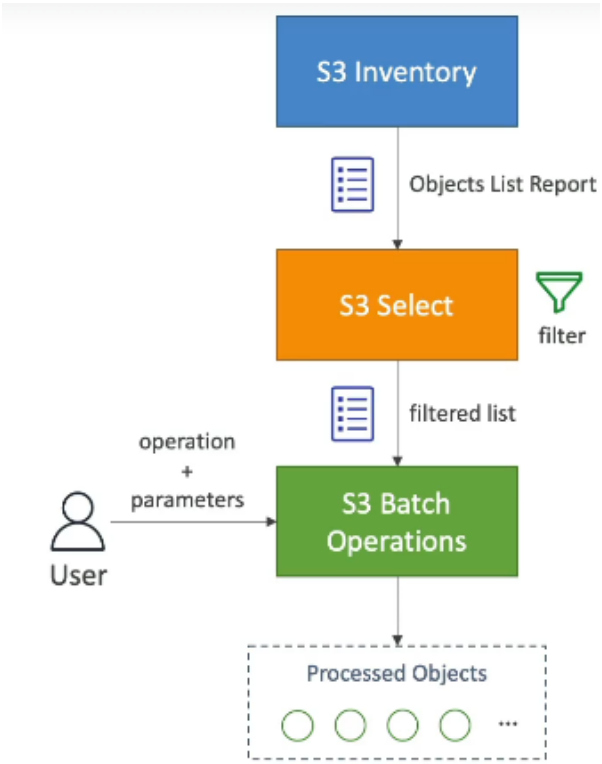


▼ ☑ S3 Batch Operations.

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.



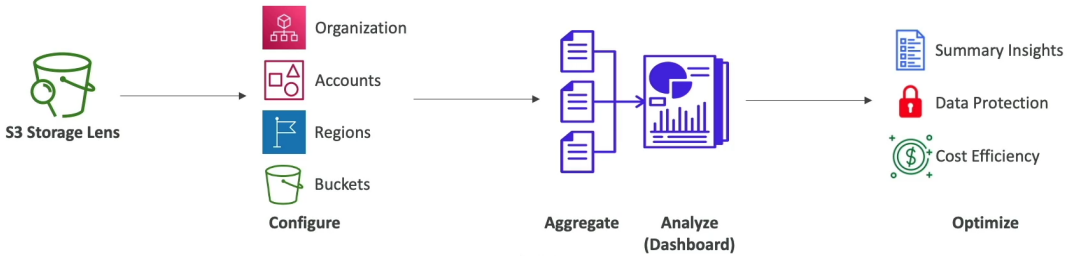
We can use S3 Inventory to get object list and use S3 Select to filter our objects.

▼ ☑ **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.

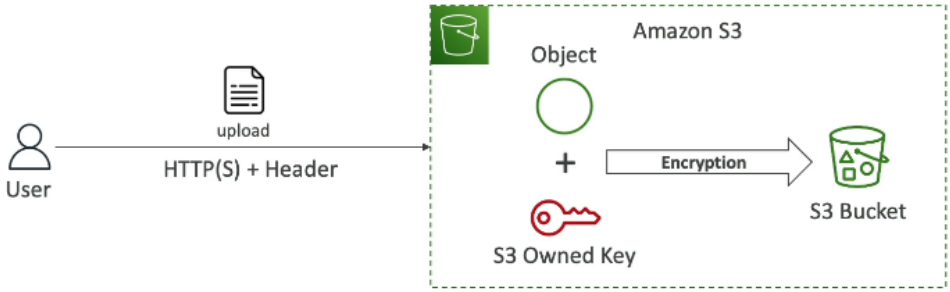
Amazon S3 Security

▼ ☑ **S3 Encryption.**

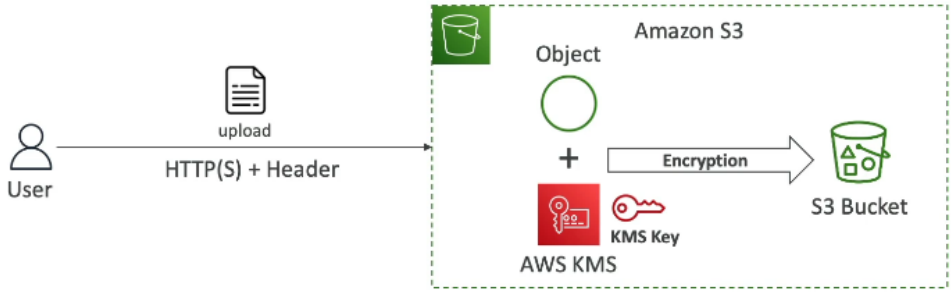
- **Server-Side Encryption (SSE)**

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

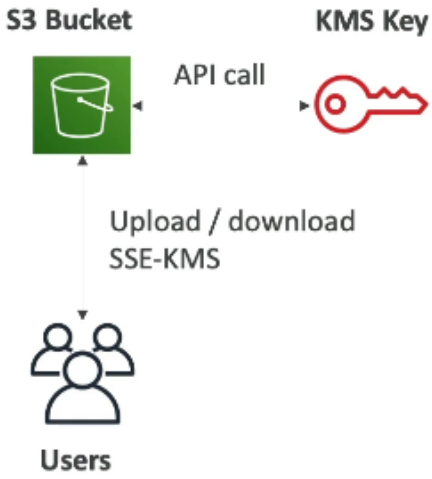
  We must set header *"x-amz-server-side-encryption":"AES256"*.



- **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"*.

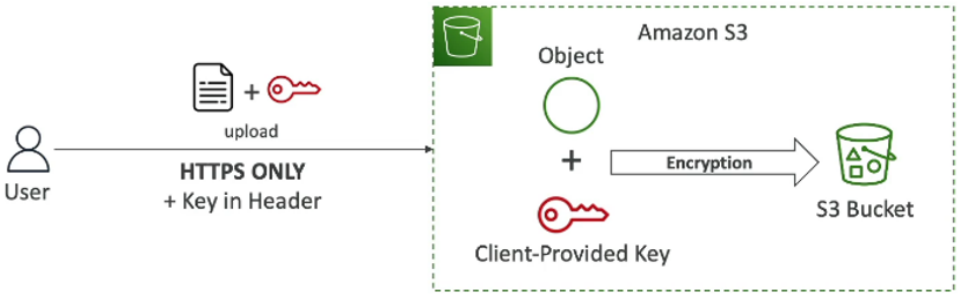


If we use SSE-KMS we may be impacted by the KMS limits. When we upload, it calls the GenerateDataKey KMS API and when we download, it calls the Decrypt KMS API.



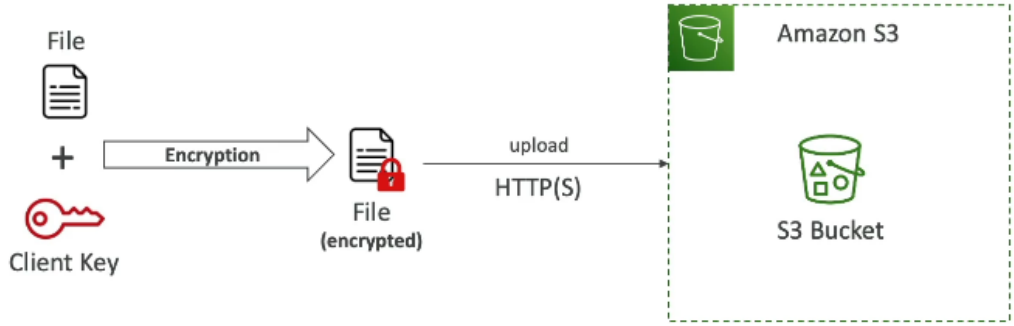
- **SSE with Customer-Provided Keys (SSE-C)** - when we want to manage our own encryption keys.

  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.



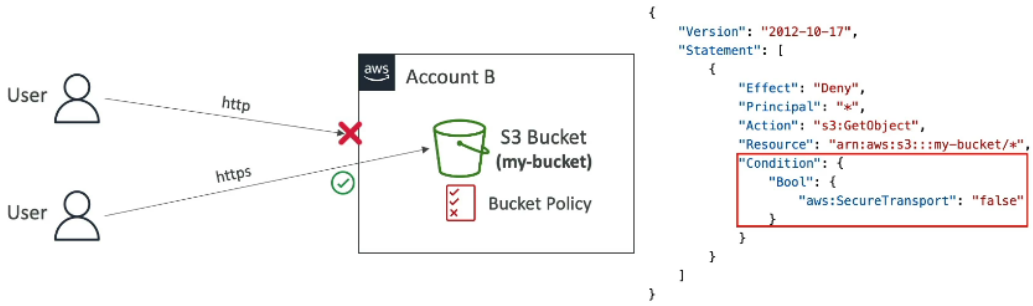
- **Encryption in Transit**

  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).

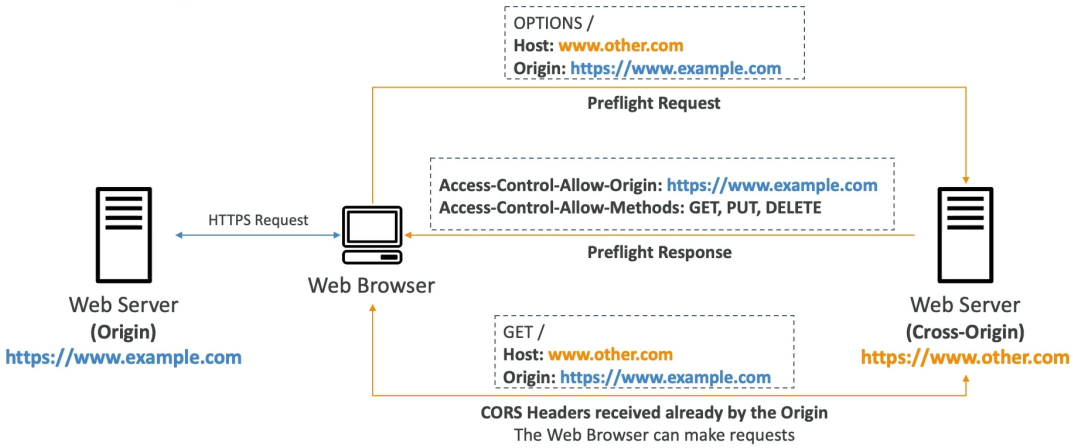
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "s3:PutObject",
            "Principal": "*",
            "Resource": "arn:aws:s3:::my-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption":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

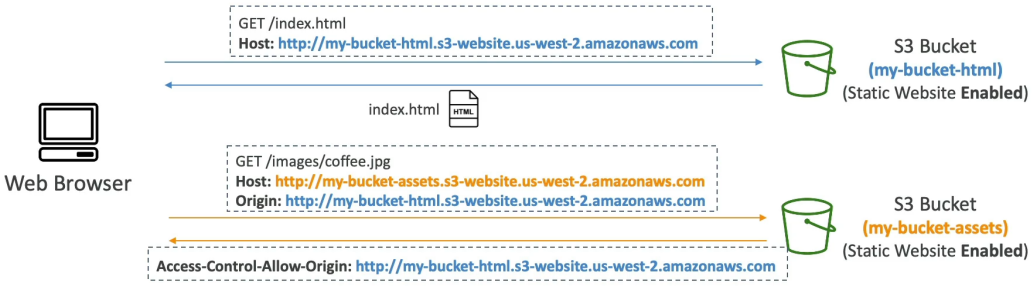
Bucket Policies are evaluated before “Default Encryption”.

▼ ☑ **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.

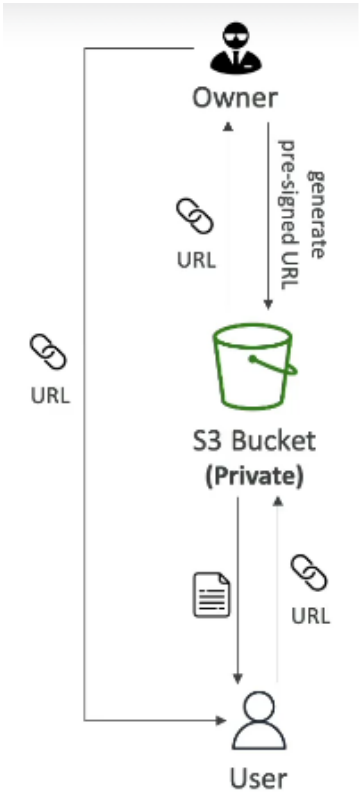
▼ ☑ S3 Pre-Signed URLs.

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**

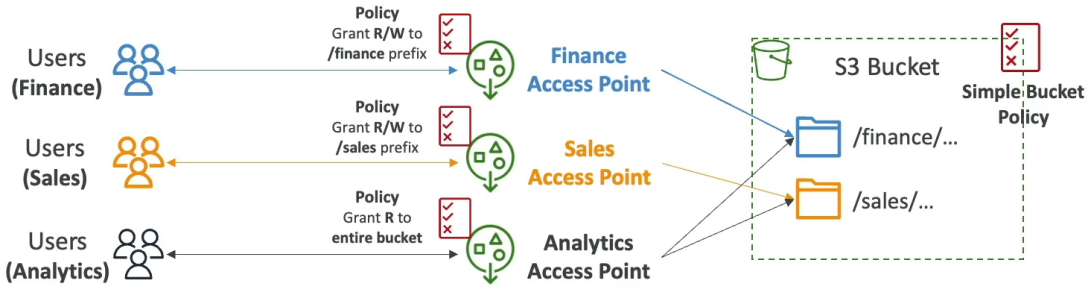
  Most users cant overwrite or delete an object version or alter its lock settings. Some users have special permissions to change the retention or delete the objects.

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

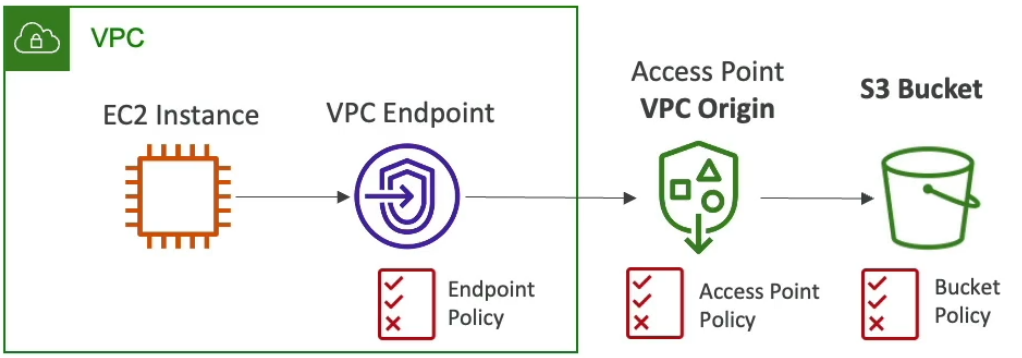
Legal Hold - protect the object idefinitely, independent from retention period.

▼ ☑ **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.




```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::awsexamplebucket1/*",
                "arn:aws:s3:us-west-2:123456789012:accesspoint/example-vpc-ap/object/*"
            ]
        }
    ]
}
```

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.