AWS

- **AWS CLI Profile**
  - The --profile CLI option helps to deal with multiple AWS accounts in CLI

```
~/.aws aws configure --profile my-other-aws-account
AWS Access Key ID [None]: Dkn86Vv9q7JE8hHn
AWS Secret Access Key [None]: hgsJgWwVR7WhRxu7z84rRAFDkPqd6hMUX2TcnPUA
Default region name [None]: us-west-2
Default output format [None]:
```

  - Here we create a profile for new account.
- **AWS CLI with MFA**
  - using MFA you take a new access-key-id and secret-access-key and a temporary token with expire data after it this credentials will be invalid
  - MFA configured from IAM
- **AWS Signature Sigv4**
  - You sign AWS HTTP request using Signature v4 **Sigv4** using two ways:
    - HTTP Header => Authorization header
    - Query String => signature in **X-Amz-Signature**
- **AWS CLI Credentials chain**
  - The CLI will look for credentials in this order
    1. Command line options
    2. Environment variables
    3. CLI Credentials file **C:\Users\.aws\credentials**
    4. CLI Configuration file **C:\Users*username*\.aws\config**
    5. Container credentials for ECS
    6. Instance profile credentials
- **SDK Credentials chain**
  - The Java SDK will look for credentials in this order
    1. Java system priorities
    2. Environment variables
    3. the default credentials profile file
    4. Amazon EC2 container credentials
    5. Instance profile credentials

- ***Important CLI credentials Scenario***
  -
  - **best practice ==> never store credentials in your code.**
- **AWS SDK**
  - Used to perform actions on AWS directly from your application code.
  - AWS CLI uses python SDK (boto3)
  - if you don't specify region when using SDK, the default region will be ***us-east-1***
- **AWS Limits (Quotas)**
  - Some AWS APIs have a call limit per seconds
  - you can request a service limit increase by **opening a ticket**
  - you can request a service Quota increase using **Service Quotas API**
  - if you get **ThrottlingExeption** intermittently use **Exponential Backoff**
  - the retry mechanism already included in SDK
  - implement retries **on 5xx** server errors and throttling **not in 4xx**
- **Regions vs Availability Zones**
  - Regions: Separate geographic areas where AWS has multiple data centers. Each region is isolated from others and designed for specific purposes like compliance and disaster recovery. Examples: "us-east-1" (North Virginia), "eu-west-1" (Ireland), "ap-southeast-2" (Sydney).
  - Availability Zones (AZs): Distinct data centers within a region, isolated in terms of power, cooling, and networking. They provide fault tolerance and high availability. Examples within a region: "us-east-1a," "us-east-1b," "us-east-1c."

IAM

- IAM = Identity and access management, Global service.
- Root account shouldn't be used or shared so we create users and group them as needed.
- Groups only contain users not other groups.
- Some users don't have to belong to a group, that is not best practice.
- User can belong to multiple groups.

- IAM : Permissions

        - Users or groups can be assigned to JSON document called o
        - policies, these policies define the permission of the users.
        - In AWS you apply the LEAST PRIVILEGE PRINCIPLE: don't give
        - more permission than a user needs.
        - Inline policy -> a policy that's only attached to a single user.

    - IAM : Password policy
        - In AWS you can setup a password policy:
            - Set minimum password length.
            - Require specific character types.
            - Allow all IAM users to change their own passwords.
            - Require users to change their passwords after some time.
            - Prevent password re-use

        - MFA (Multi Factor Authentication)
            - MFA = password + security device you own.
            - Even if password is stolen the account is not compromised.
            - MFA devices options:
                - Virtual MFA devices:
                    - Google Authenticator - Authy
                - Universal 2nd Factor (U2F) Security Key:
                    - physical device (3rd party)
                        - YubiKey from Yubico.
                - Hardware Key Fob:
                    - Gemalto - 3rd party
                - Hardware Key Fob For AWS GovCloud:
                    - SurePassID - 3rd party

    - Ways of Accessing AWS
        - AWS Management Consol.
        - AWS CLI (Access Key ID and Secret Access Key).
            - A tool enables you to interact with AWS using commands in command line shell.
        - AWS SDK (Access Key ID and Secret Access Key).
            - Language-Specific APIs (set of libraries), access programmatically.
        - Using cloud shell is available in some regions.

        - Its like using CLI but in AWS cloud or consol.
        - It has some features like upload and download files.

    - IAM Roles
        - Some AWS service will need to perform some actions on your behalf and need permission.
        - To do so we will assign permissions to AWS with IAM Roles.

    - IAM Security Tools
        - IAM Credentials Report (account-level)
            - a csv report lists all your account's users and the status of their various credentials.
        - IAM access Advisor (user-level)
            - list the permissions granted to the user and when each service last access.
            - it helps when you want to remove the services that user doesn't need in order to have LEAST PRIVILAGE PRINCIPLE applied.

EC2

- EC2 = Elastic Compute Cloud -> infrastructure as a service.
- Consist of :
    - Renting virtual machines (EC2)
    - Storing data on virtual drives (EBS)
    - Distributing load across machines (ELB)
    - Scaling the services using an auto-scaling group (ASG)
- Sizing and configuration options:
    - OS -> Linux, Windows and Mac
    - How much compute power & cores (CPU)
    - How much RAM
    - How much storage space:
        - Network attached -> EBS & EFS.
        - Hardware (EC2 instance store).
    - Network card -> card speed and Public IP.
    - Firewall rules : security group.

        - Bootstrap script (configure at first launch): EC2 User Data.

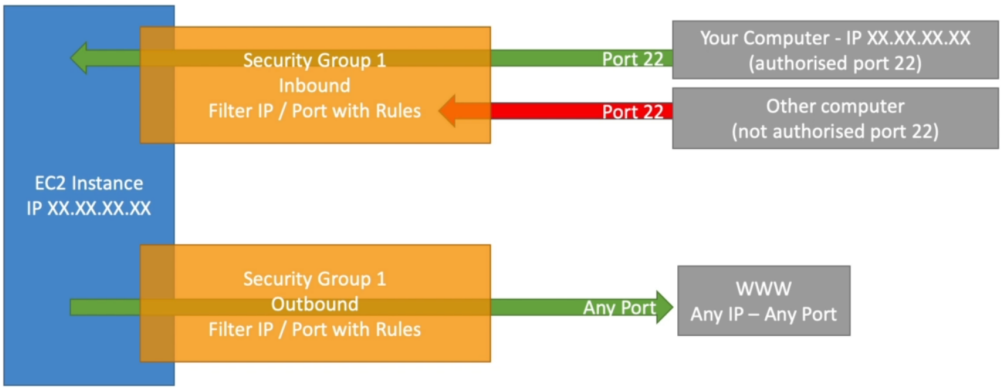
- EC2 User Data
    - Bootstrap the instance using EC2 User data script.
    - bootstrapping -> launch commands when a machine starts.
    - the script run once at the instance first start.
    - script run as administrator.
    - EC2 User Data used to automate boot tasks :
        - Installing updates.
        - Installing software.
        - Downloading common files from the internet.
        - Anything else.

- Some EC2 configuring tips:
    - m5.2xlarge what this mean?
        - m -> instance class.
        - 5 -> generation, AWS improve them over time.
        - 2xlarge -> size starts with small and so.
    - use .pem key pair for Linux, MAC and W10
    - use .ppk for Windows less than 10
    - When you restart the instance the public key changes but the private not.

- EC2 types
    1. General Purpose
        - Great for diversity of workloads such as web servers or code repositories.
        - Balance between Compute, Memory and Networking.
    2. Compute Optimized
        - Great for compute-intensive tasks that require high performance processors.
            - Batch processing workloads
            - Media transcoding
            - High performance web servers
            - High performance computing (HPC)
            - Scientific modeling & Machine learning
            - Dedicated gaming servers

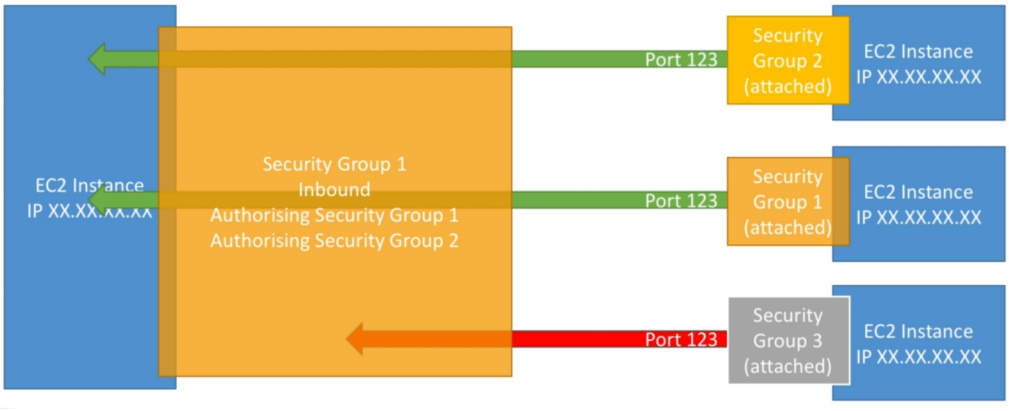
3. Memory Optimized
    - Fast performance workloads that process large data in memory
        - High performance relational/non-relational databases
        - Distributed web scale cache stores
        - In-memory databases optimized for BI
        - Application performing real-time processing.
4. Storage Optimized
    - Great for storage-intensive tasks that require high, sequential read and write access to large data sets on local storage.
        - High frequency online transaction processing systems (OLTP)
        - Relational/non-relational databases.
        - Cache for in-memory databases
        - Data warehousing applications
        - Distributed file systems

- Security Groups
    - Control the how traffic is allowed into or out of our EC2 instance.
    - Acting as firewall on EC2 instances
    - Regulate access to ports, authorized IP ranges, IPv4 and IPv6
    - Control inbound -> from others to instance
    - Control outbound -> from instance to others



    - Can be attached to multiple instances
    - Locked down to a region -> if you change the region then you must create new one.
    - It's surround the EC2 block or allow the traffic.
    - Good to maintain one separate security group for each SSH access
    - If your application not accessible (time out) then its security group issue.

- If your application gives a connection refused error then its an app error or something out of security group.
- By default all inbound traffic is blocked but outbound is allowed.
- referencing other security groups



- Classic important Ports to know
  - 22 SSH (secure shell) -> log into Linux instance
  - 21 FTP (File Transfer Protocol) -> upload files into a file share
  - 22 SFTP (Secure File Transfer Protocol) -> upload files using SSH
  - 80 HTTP -> access unsecure websites
  - 443 HTTPS -> access secured websites
  - 3389 RDP (Remote Desktop Protocol) -> log in windows instance

- EC2 instances purchasing options
  - EC2 On-Demand
    - pay for what you use
      - Linux or Windows -> billing per second
      - Other OSs -> billing per hour
    - Has highest cost - no upfront payment
    - No long-term commitment
    - Recommended for short-time and un-interrupted workloads where you can't predict how the application will behave.

  - EC2 reserved instances
    - up to 72% discount compared to On-Demand
    - you reserve (instance type - region - tenancy - OS)

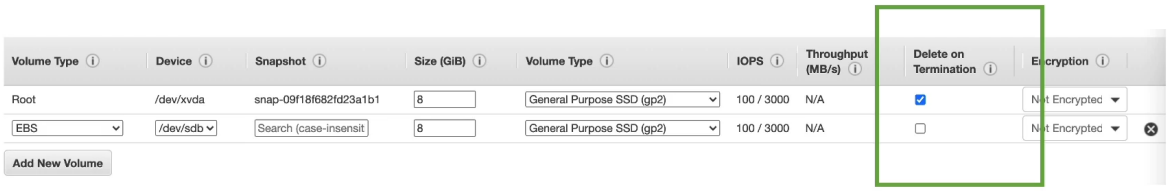
        - 1 year (+discount) or 3 years (+++discount)
        - scope regional or zonal
        - recommended for steady states applications (database for example)
        - you can buy and sell in the reserved marketplace

        - Convertible reserved instance
            - type of reserved instances
            - can change the EC2 instance type, instance family, OS, scope and tenancy.
            - but less discount cause of flexibility up to 66%
- EC2 Saving plans
    - discount based on long-term usage up to 72%
    - commit to a certain type of usage ($10/h for 1 or 3 years)
    - billed at the on-demand price
    - locked to a specific instance family & region (M5 in us-east-1)
    - flexible across
        - instance size (m5.xlarge, m5.2xlarge...)
        - OS
        - tenancy (Host, Dedicated, Default)
- EC2 spot instances
    - can get discount up to 90%
    - but you can lose the instance at any point of time if your max price is less than the current spot price
    - most cost efficient instance in AWS
    - useful for workloads that are resilient to failure
        - Batch jobs
        - Data analysis
        - Image processing
        - Any distributed workloads
        - Workloads with flexible start and end time
- EC2 dedicated hosts
    - A physical server with EC2 instance capacity fully dedicated to your use.
    - gives you visibility into lower level hardware.
    - Purchasing options
        - on-demand - pay per second

        - reserved 1 or 3 years
        - most expensive
    - useful for software that have complicated licensing model (BYOL - Bring your own license)
    - for companies that have strong regulatory or compliance needs.
- EC2 Dedicated instance
    - instances run on hardware that dedicated to you
    - may share hardware with other instances in same account
    - no control over instance placement (can move hardware after stop/start)

- EC2 Capacity reservation
    - reserve EC2 capacity in a specific AZ for any duration
    - always have access to EC2 capacity when you need it
    - no time commitment (create/cancel any time) no billing discount
    - you are charged at on-demand whether you run instances or not
    - suitable for short-term uninterrupted workloads that needs to be in a specific AZ.

- EC2 storage
    - EBS (Elastic Block Store)
        - a drive you can attach to your instance while they run through the network
        - allows your instances tp persist data even after termination
        - can only mount to one instance at a time.
        - bound to a specific availability zone.
            - so if EC2 in us-east-1a can not attach to EBS in us-east-1b but you can take a snapshot to move it.
        - have provisioned capacity (size and IOPS) and you can increase it.
        - By default the root EBS volume is deleted when EC2 terminates (attribute enabled) and any other attached EBS is not deleted (attribute disabled) but this can be controlled



- EBS Volume types
  - General Purpose SSD
    - gp2/gp3 SSD (1 GiB - 16 GiB)
      - general purpose , cost effective and low-latency
      - 3,000 up to 16,000 IOPS
  - Provisioned IOPS SSD
    - for critical business apps or apps need more than 16,000 IOPS
    - Great for database workloads
    - io1/io2 (4 GiB - 16 TiB)
      - 64,000 PIOPS for Nitro EC2 and 32,000 for others
      - can increase PIOPS independently from size
    - io2 Block Express (4 GiB - 64 GiB)
      - 256,000 PIOPS
  - Hard Disk Drive (HDD)
    - cannot be a boot volume
    - throughput optimized HDD (st1)
      - Big Data, Data warehousing or long processing
      - max IOPS 500 - max throughput 500 MiB/s
    - cold HDD (sc1)
      - for data that is infrequently accessed
      - where lowest cost is important
      - max IOPS 250 - max throughput 250 MiB/s

  - EBS multi attach
    - only (io1) and (io2) family has this feature
    - attach the same EBS for multi EC2 instances in the same AZ at the same time
    - up to 16 EC2 instance at a time
- Snapshots

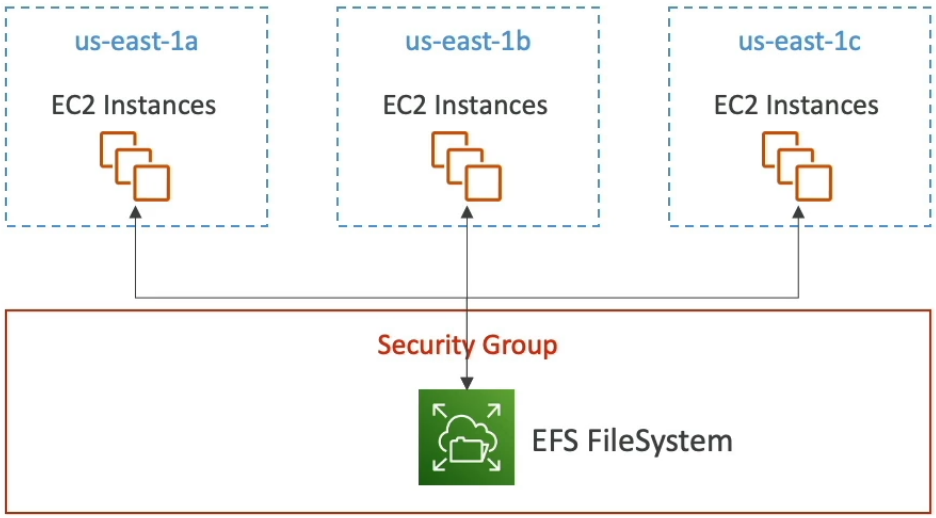
        - snapshot is a backup for your EBS volume at any point of time.
        - its recommended to do snapshot before detach volume.
        - snapshot used also to copy EBS across AZ.

        - EBS snapshot archive
            - move snapshot to archive tier is 75% cheaper.
            - takes within 24 to 72 hours for restoring the archive.
        - Recycle Bin for EBS snapshots
            - retain deleted snapshots so you can cover them after accidental deletion.
            - you can decide the retention to be from a day to a year as you want.
        - Fast snapshot restore
            - force full initialization of snapshots to have no latency on the first use.

- AMI (Amazon Machine Image)
    - a customization of an EC2 instance
    - add your software and configurations and then create an AMI from this instance and reuse it as you need.
    - faster boot because all your software is pre-packaged and can be copied across regions.

- EC2 instance store
    - if you want alternative to EBS cause of its network latency or its another limitations or you want high performance hardware disk then use EC2 instance store.
    - better I/o performance
    - but EC2 instance store lose their storage if they are stopped or terminates
    - Good for buffer/ cache/ scratch temporary content
    - risk for data lose if hardware fails

- EFS
    - network file system (NFS) that can be mounted to many EC2
    - EFS works with EC2 in multi -AZ

        - highly available, scalable, expensive (3 x gp2 price) but Pay per use
        - 

        - use cases: content management, web serving, data sharing and wordpress
        - uses NFSv4.1 protocol
        - compatible with Linux based AMI not windows
        - scale automatically pay-per-use

        - performance
            - General purpose (default)
                - latency sensitive use cases -> web servers
            - MAX I/O
                - higher latency, throughput and parallel -> big data and media processing

- EC2 Instance Metadata (IMDS)
    - Allows AWS EC2 to learn about itself (IP, hostname,...etc)
    - It access itself through http://169.254.169.254/latest/meta-data
    - You can get the IAM role name from this metadata but you cannot get the IAM policies
    - **IMDSv1 VS IMDSv2**
        - IMDSv1 => access the URL above directly
        - IMDSv2 =>more secure, get session token then use it to access the metadata

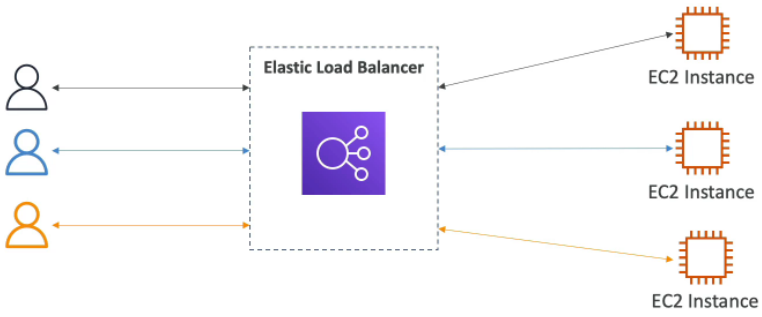
---

Elastic Load Balancing

Scalability and Availability

- scalability
  - means that an application / system can handle greater loads by adapting.
  - vertical and horizontal
    - vertical
      - increasing the size of the instance -> t2.micro to t2.large
      - very common for non-distributed systems such as a database
      - there was a limit for how much you can vertically scale (Hardware limit)
    - horizontal
      - increasing the number of instances / system for your application
      - very common for modern application
- availability
  - means running your application in at least 2 data centers (availability zones)
  - goes hand to hand with horizontal scaling.
  - can be active or passive.

- Load balancer
  - servers that forward traffic to multiple servers (EC2 instances)

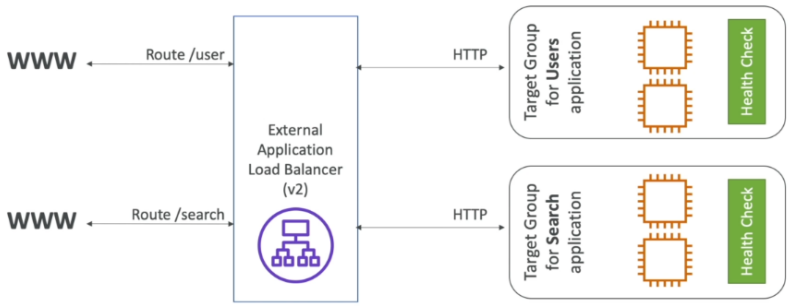


  - spread load across multiple downstream instances
  - handle failures of down stream instances

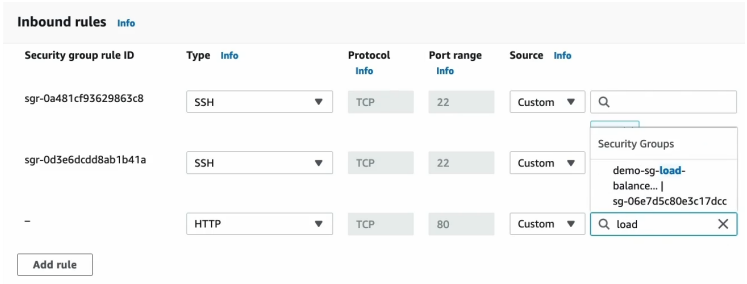
- do regular health check
- provide SSL termination

- Elastic load balancer
  - AWS guarantees that it will be working
  - AWS takes care of upgrade, maintenance, high availability
  - AWS provides only few configuration knobs
  - ELB integrate with most AWS services
  - it costs less to setup your load balancer but it will be a lot more effort on your end

  - **4 load balancers**
    - **Classic load balancer**
      - The Classic Load Balancer is deprecated at AWS and will soon not be available in the AWS Console.
    - **Application load balancer**
      - layer 7 (http)
      - load balancing to multiple applications across machines (target groups) or on the same machine (containers)
      - support HTTP/2 and WebSocket
      - support redirects from http to https
      - routing table to different target groups based on path in URL or hostname or QueryString
      - great fit for microservices and container based application



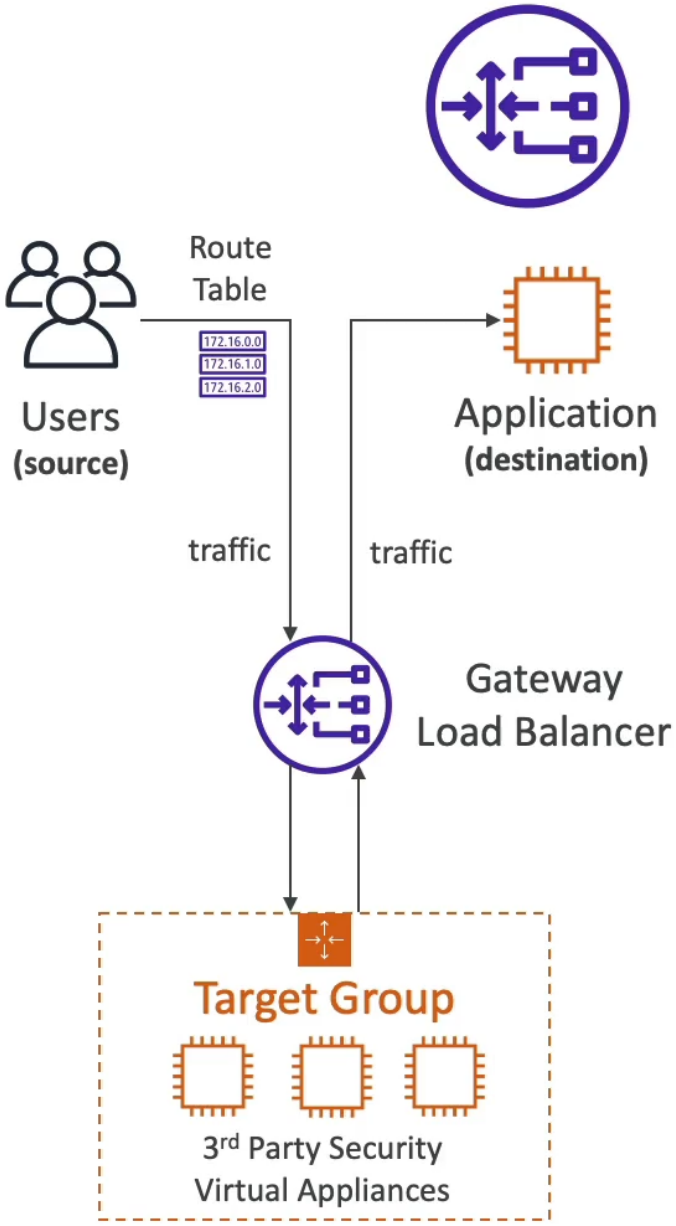
      -
      -
      - ALB also supports Lambda functions as targets, allowing you to create powerful serverless application.

- the application server don't see the IP of the client directly because its replaced with Load balancer IP but it inserted into the header in X-Forward-For and the port in X-Forward-Port and porto in X-Forward-Porto.
- for security its better to make only the load balancer ip is acceptable for instances by changing there security group by adding the load balancer security group in inbound role http source
- security group
  - load balancer allow http/https from anywhere
  - application (EC2) allow traffic only from load balancer
  - 

- **Network load balancer**
  - NLB operates at Layer 4 (Transport Layer) of the OSI model.
  - It distributes traffic **based on IP** protocol data (source IP, destination IP, source port, destination port).
  - High Availability: Spans multiple Availability Zones for fault tolerance.
  - Static IP: **Provides a fixed IP (one or more IP)** address for the load balancer.
  - Health Checks: Support HTTP, HTTPS and TCP for health checking.
  - Connection Handling: Handles millions of simultaneous connections with low latency.
  - Port Forwarding: Supports multiple ports on the same IP address.

- Integration and Configuration:
  - Can be integrated with Auto Scaling, AWS Certificate Manager, and other AWS services.
  - Configured using listeners, rules, and target groups.
- Use Cases:
  - Ideal for applications needing high availability and scalability.
  - Suited for TCP/UDP-based applications.
- NLB is generally more expensive than Application Load Balancers (ALBs) due to its capabilities.

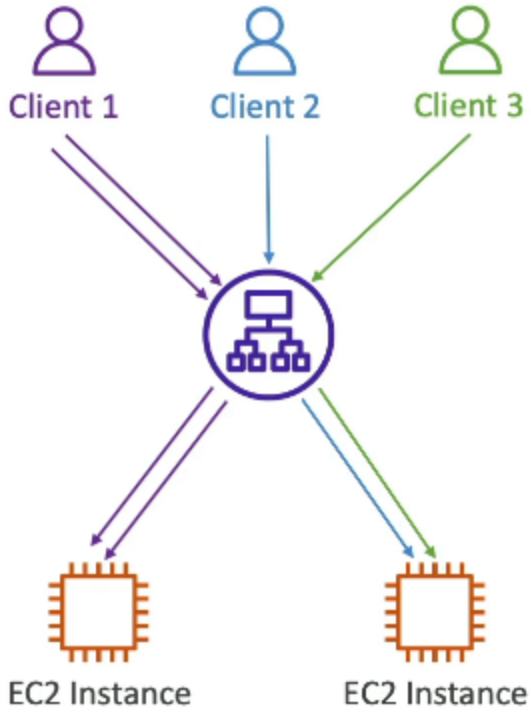
- **Gateway load balancer**
  - AWS Gateway Load Balancer (GWLB) is designed for distributing traffic across virtual appliances like firewalls and security systems then the traffic is passed from appliances to the GWLB again then it distribute the traffic to application instances.



-
- Operate at layer 3
- Routes incoming and outgoing traffic through a fleet of virtual appliances.
- Uses GENEVE protocol on port 6081

- Sticky Sessions (Session Affinity)

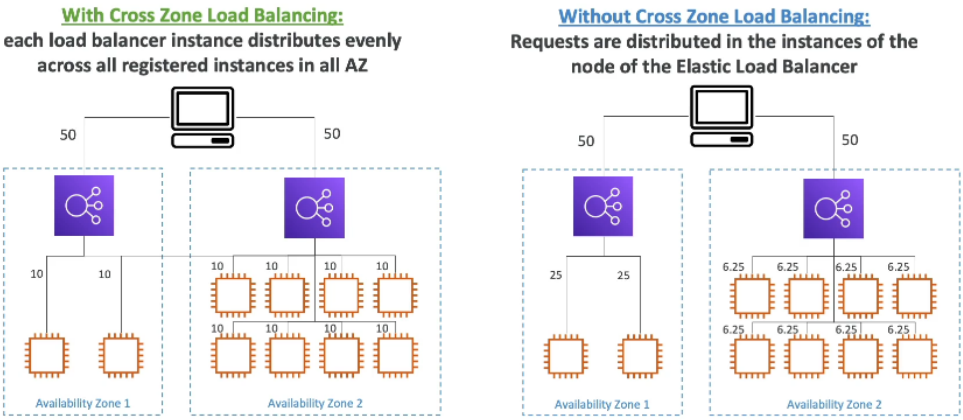
  When implementing stickiness the same client is always redirected to the same instance behind a load balancer



-
- works with CLB, ALB, NLB and ELB with auto scalling
- it use cookies to do that and cookies has an expiration date you can control it.
- enabling stickiness may bring imbalance to the load over the EC2
- use case: when dealing with stateful apps we and want to make sure that user doesn't lose his session data

- Cross-Zone Load Balancer

  In case you have 10 instances in AZ and 2 in other AZ then the Cross-Zone Load Balancer is designed to distribute incoming traffic across multiple Amazon Web Services (AWS) Availability Zones. It's used to enhance availability, fault tolerance, and even distribution of workloads, it deals with different AZs as a single AZ.

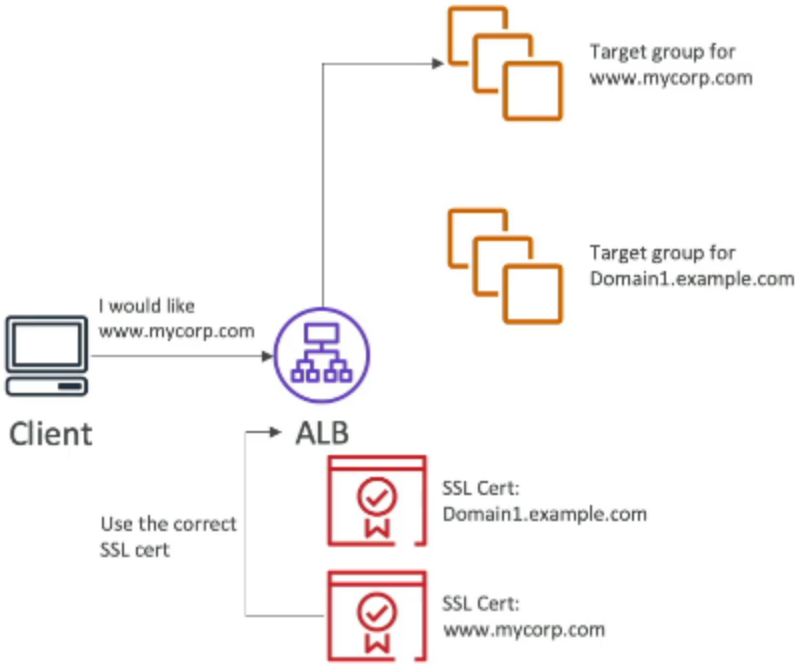
- 

  - Cost Implications: While Cross-Zone Load Balancing provides benefits, distributing traffic across multiple AZs might involve additional data transfer costs. Be mindful of this in your cost management strategy.
  - in ALB the Cross-Zone Load Balancer is enabled by default so no addition money is added and you can disable it by editing the target group attributes.
  - in NLB the Cross-Zone Load Balancer is disabled by default so addition money is added by enabling it and you can enable it by editing the NLB attributes.
  - in CLB the Cross-Zone Load Balancer is disabled by default but no addition money is added by enabling it and you can enable it by editing the CLB attributes.

- SSL/TLS Certificates
  - SSL (Secure Socket Layer)allows traffic between clients and Load Balancer to be encrypted in transit (in-flight encryption)
  - TLS is a newer version of SSL
  - SSL have an expiration date and must be renewed
  - You can manage SSL certificate from ACM (AWS Certificate Manager)or upload your own
    - SNI (Server Name Indication)

      SNI stands for Server Name Indication, and it's an extension to the TLS (Transport Layer Security) protocol that allows multiple SSL/TLS certificates to be hosted on the same IP address and port combination. This is particularly useful in scenarios where

a single IP address needs to serve multiple domains with different SSL certificates.

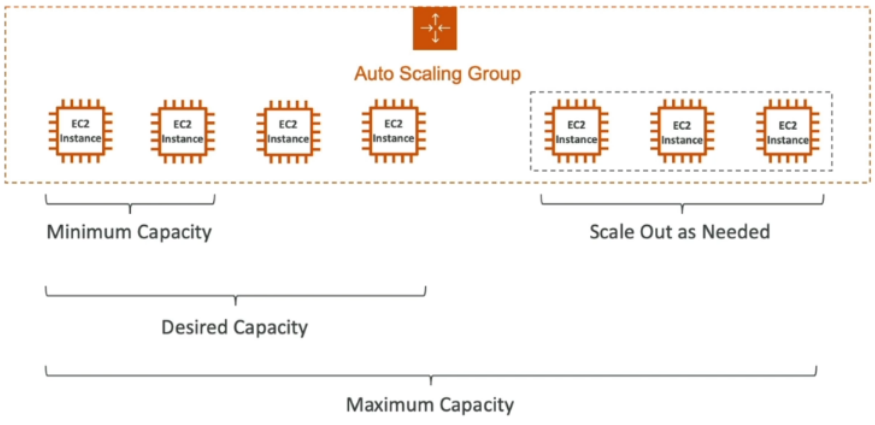


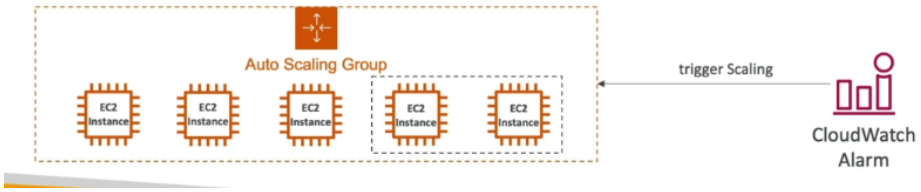
With ELB, you can use SNI to associate multiple SSL/TLS certificates with different domains. This allows the load balancer to route traffic based on the SNI header sent by the client, enabling it to select the appropriate certificate for the requested domain.

Only work with ALB, NLB and CloudFront and doesn't work with CLB.

- Connection Draining (Deregistration Delay)
  a feature provided by Elastic Load Balancing (ELB) in Amazon Web Services (AWS). It allows an Amazon ELB load balancer to stop sending new requests to instances that are being taken out of service (e.g., for scaling purposes or instance maintenance) while still allowing in-flight (current) requests to complete. This ensures a smooth transition without abruptly cutting off active user sessions.

1. **Enabled Duration:** When you enable Connection Draining, you set an "enabled duration" that specifies the time during which the load balancer allows in-flight requests to complete before stopping new requests to an instance.
2. **In-Flight Requests:** During the enabled duration, in-flight requests are allowed to complete on instances being taken out of service. This prevents abrupt termination of active user sessions.
3. **Completion of Requests:** After the enabled duration, the load balancer initiates the connection draining process, ensuring that all ongoing requests are finished before removing instances from rotation.
4. **Instance Removal:** Once connection draining is complete, instances are removed from the load balancer's rotation, preventing new requests from being sent to those instances.
5. **Sudden Termination:** Without Connection Draining, terminating instances might lead to the abrupt termination of active user sessions, causing user experience issues.

- ASG (Auto Scaling Group)
  - Scale Out (add EC2 instances) to match an increased load
  - Scale In (remove EC2 instances) to match a decreased load
  - Automatically add new instances to the load balancer
  - Re-create an EC2 instance in case of unhealthy case happened
  - ASG is free you only pay for new instances
  - 

- You need to specify a Launch Template which include all the system configuration (ELB, EC2, SG,...)
- Specify Min, Desired(Initial) and Max instances Capacity
- CloudWatch Alarms
  - its possible to scale(OUT-IN) based on alarms
  - the alarm monitor metrics (Average CPU,....)
  - 

- Dynamic scaling policies
  - Target Tracking Scaling
    - most simple and easy to set-up
    - ex: i want the average of ASG CPU to stay around 40%
  - Simple/Step Scaling
    - When a CloudWatch alarm is triggered (example: CPU > 70%) then add 2 units
    - When a CloudWatch alarm is triggered (example: CPU < 30%) then remove 1 units
  - Scheduled Actions
    - Anticipate a scaling based on known usage pattern
    - ex: increase the min capacity to 10 at 5pm on Fridays
- Predictive Scaling
  - continuously forecast load and schedule scaling ahead
  - depend someway on AI

- Good metrics to scale on
  - CPU
  - Requests count
  - Network IN/OUT
  - custom metric by CloudWatch

- Scaling Cooldowns

        - after scaling activity happens you are in cooldown period by default 300 seconds
        - during this cooldown period the ASG will not add or remove instances to allow for metrics to stabilize first
        - its better to use AMI to reduce configuration time.

    - Instance Refresh
        - to re-create all instances with new template.

RDS (Relational Database Service)

Allows you to create databases in the cloud that are managed by AWS

- Postgres, MySQL, MariaDB, Oracle, MSSQL and Aurora (AWS proprietary database)

- Why using RDS over deploying DB on EC2 ??
    - Automated Provisioning and OS patching
    - Continuous backups and point in time restore
    - Monitoring dashboards
    - Read replicas for improved read performance
    - Multi AZ for Disaster Recovery
    - Maintenance windows for upgrades
    - Scaling capability (Vertical and Horizontal)
    - Storage backed by EBS (gp2 or Io1)

- SSH into RDS is not available

- Storage Auto Scaling
    - increase RDS DB dynamically
    - when RDS detect that you are running out of free database storage it scales automatically
    - avoiding manual scaling
    - you have to put Maximum Storage Threshold (maximum limit for DB storage)
    - automatically modify storage if:

36. free storage is less than 10% of allocated storage
37. low-storage lasts at least 5 minutes
38. 6 hours have passed since last modification

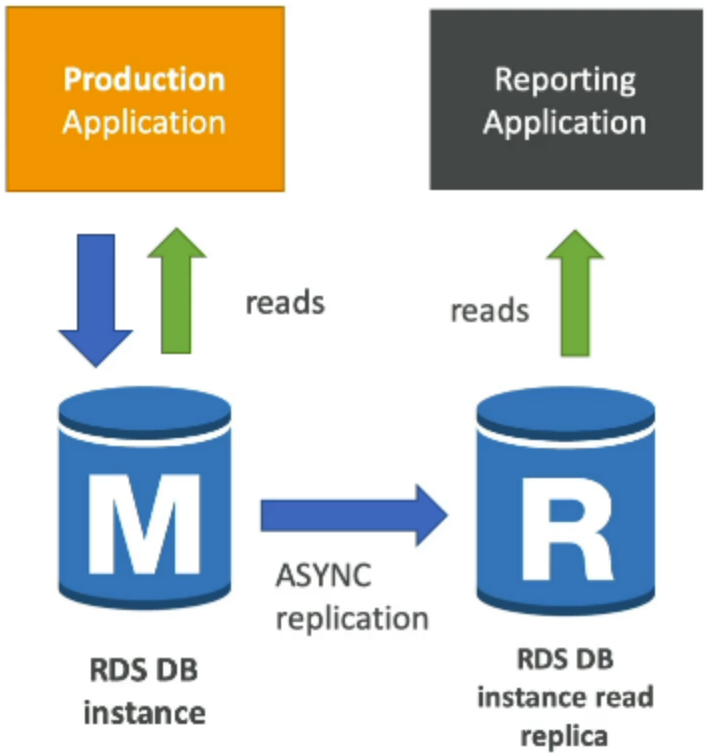
- useful for an applications with unpredictable workload

- Read Replicas
  - RDS read replicas are a feature of Amazon RDS that allows you to create one or more copies of your primary (master) database instance. These read replicas are read-only copies of the primary database that can be used to offload read traffic from the primary instance, improve scalability, and enhance performance.
  - Up to 15 Read Replicas within AZ, cross AZ or cross Region

  - **Read Replicas are eventually consistent:**
    1. Write Operations: Changes made on the primary database are immediately applied there. However, there's a delay before these changes are propagated to read replicas.
    2. Read Operations: Read replicas might not immediately reflect the most recent changes from the primary instance. There could be a brief period where read data is slightly outdated.
    3. Replication Lag: The time delay between primary writes and read replica updates is called "replication lag." Factors like network latency and write volume influence this lag.

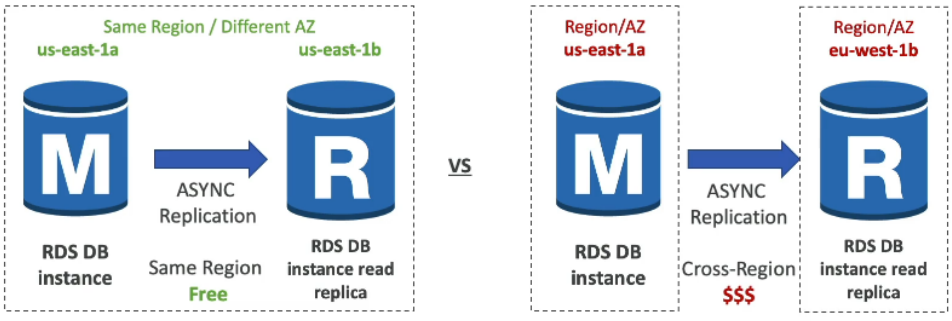
  - Read replicas can be promoted to become their own standalone database instances. When a read replica is promoted, it transitions from being a read-only replica to a fully functional primary (master) database

  - Use Case
    - in case you want to run a reporting application to do some data analysis and to prevent this from effecting the master database performance you create a Read Replica



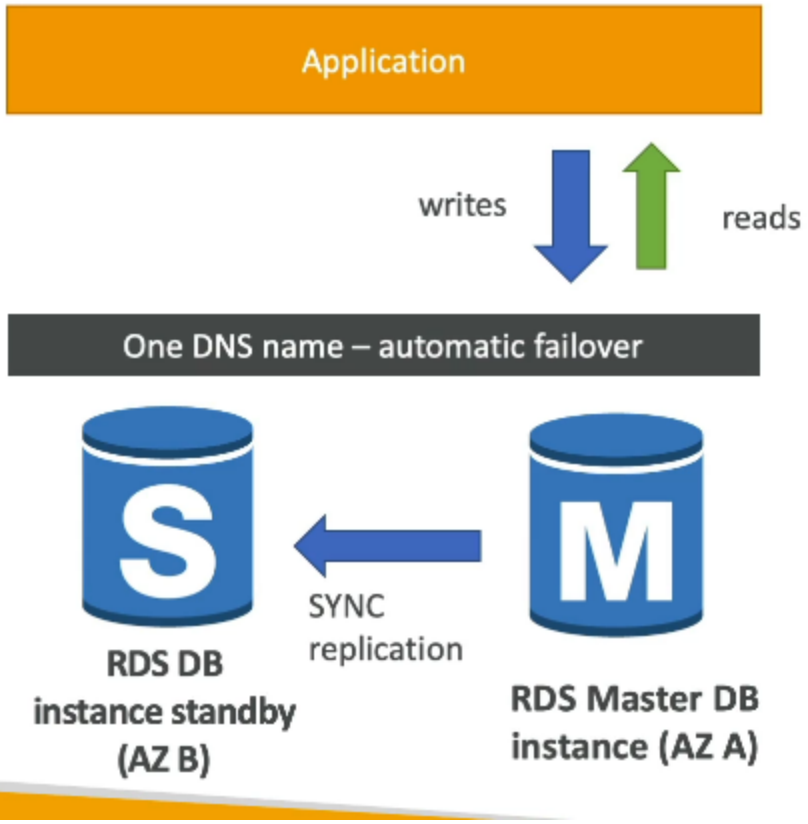
- Note that Read Replicas connection with master is ASYNC
- Read Replicas can be used as multi AZ DR

- Read Replicas Network cost
  - In AWS there is a network cost when you goes from one AZ to another but for RDS Read Replicas no network cost in same region and different AZ but there is in different region



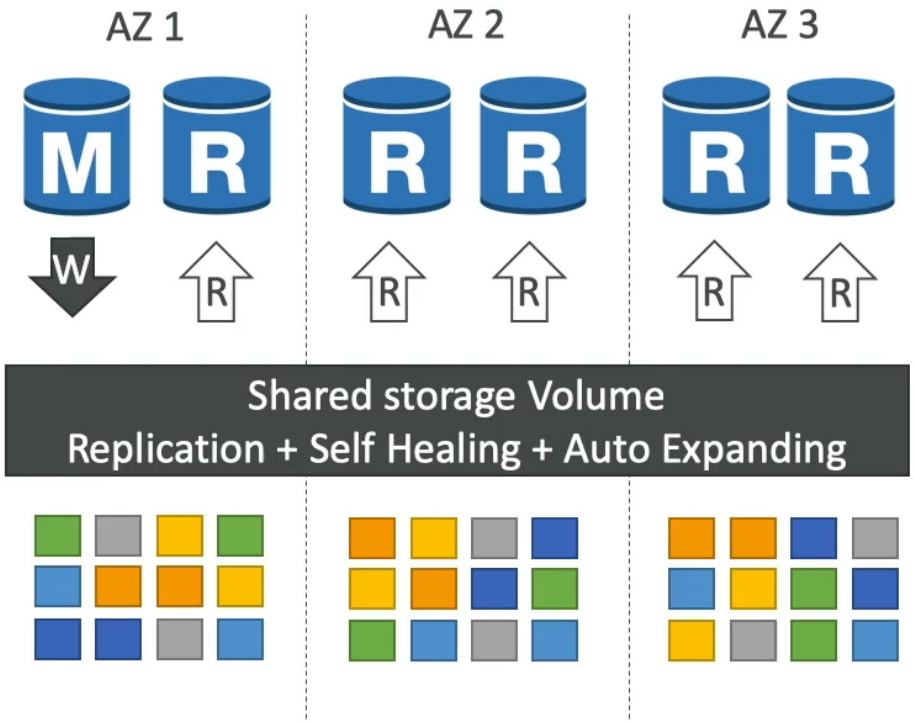
- Multi AZ Disaster Recovery
  - Another standby RDS instance for failure cases
  - Master and Recovery RDS are with one DNS name for automatic failover

- Failover in case of loss of AZ , network, instance or storage failure
- No manual intervention
- Not used for scaling
- Read Replicas can be used as multi AZ DR



- Amazon Aurora
  - Amazon Aurora is a relational database engine developed by Amazon Web Services (AWS). It is designed to provide high performance and availability while being compatible with MySQL and PostgreSQL databases.
  - Aurora storage automatically grows in increments of 10GB to 128GB and can have up to 15 replicas.
  - **Automatic Failover**: Aurora provides automatic failover (less than 30 second) in case the primary instance becomes unavailable. It promotes a read replica to become the new primary, minimizing downtime.

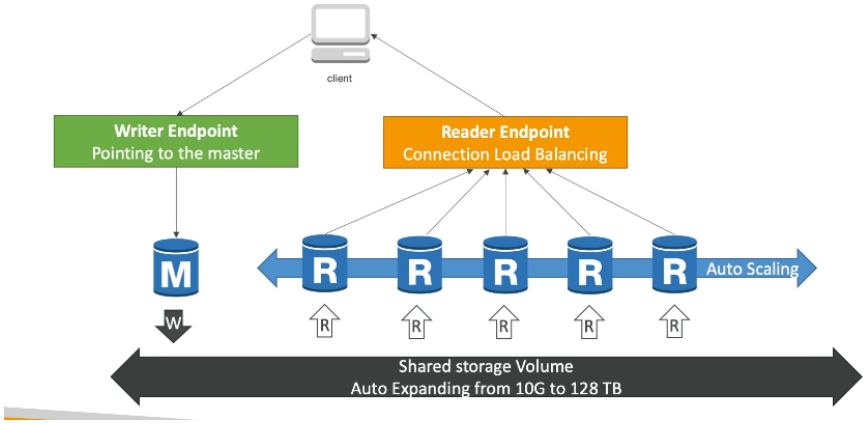
- While Aurora provides enhanced performance and features, it may have a higher cost(about 20% more) compared to traditional RDS instances due to its advanced capabilities.
- Aurora's data is automatically replicated across multiple Availability Zones (AZs), providing high availability and fault tolerance.
- 6 copies of data across 3AZ, 4 needed for writes and 3 for reads



- **Writer Endpoint and Reader Endpoint**
  - **Writer Endpoint**
    1. The writer endpoint is the **primary endpoint** for your Aurora database cluster.
    2. It points to the primary instance of the cluster where write operations are directed (INSERT, UPDATE, DELETE).
    3. All changes to the database should be made through the writer endpoint to maintain data consistency.
    4. During a failover event, Aurora automatically promotes a read replica to become the new primary. The writer endpoint is updated to point to the new primary instance.

5. Writer endpoint can also be used to perform read operations (SELECT queries) when necessary.

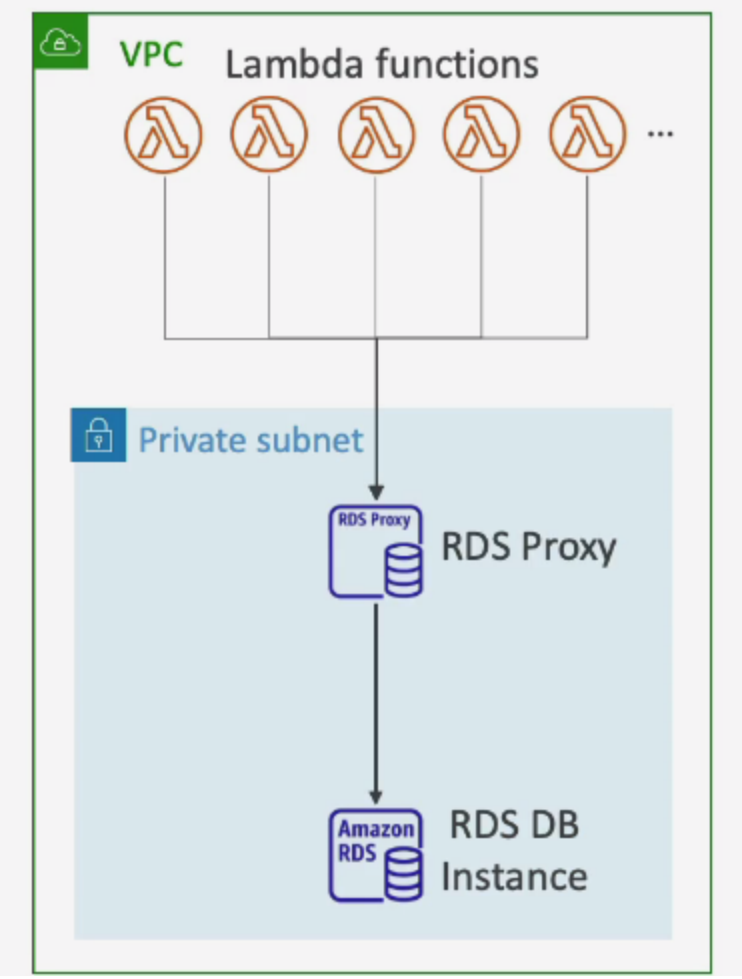
- **Reader Endpoint**
  80. The reader endpoint is an optional endpoint associated with an Aurora database cluster **for only read purposes.**
  81. It serves as a single endpoint for read traffic and can distribute that traffic across multiple read replicas.
  82. Read replicas are additional instances that replicate data from the primary instance.
  83. The reader endpoint helps offload read traffic from the primary instance, improving overall performance.
  84. Read replicas can be used to horizontally scale read-heavy workloads and improve query performance.
  85. The reader endpoint abstracts the underlying read replicas, allowing your application to switch between them for high availability and load balancing.
  86. The reader endpoint can also facilitate cross-region read traffic in Aurora Global Databases if enabled.



- In Amazon Aurora, both the master instance (writer) and read replicas share the same underlying storage volume. This shared storage architecture is a fundamental aspect of how Aurora manages data replication and provides high availability and performance while the storage in traditional RDS and replicas is not shared.

- RDS and Aurora provides encryption at rest using AWS Key Management Service (KMS) and supports TLS encryption for data in transit (in-flight encryption, Ready by default) using AWS TLS, ensuring the security of your data.
- Can use IAM Authentication to connect to the database instead of username and password.
- Controlling Network access using security group.
- No SSH available except on RDS custom service
- Audit Logs can be enabled and sent to CloudWatch Logs.

- Amazon RDS Proxy
  - RDS Proxy manages database connections on behalf of your application. It maintains a connection pool and helps reduce the overhead of opening and closing connections to the database. it acts as an intermediary between your application and your RDS databases, and open a connection with the database and connect all the application using only this connection.
  - **Scalability**: RDS Proxy enables your application to handle a large number of database connections by sharing a smaller number of connections with the database instances. This can improve the overall scalability of your application.
  - **Caching:** RDS Proxy can cache frequently used queries, reducing the load on the database instances and improving query response times.
  - **Failover Handling:** Reducing failover time (by up 66%) because instead of reconnect all applications again to the RDS now you have only one connection to reconnect which is the RDS Proxy connection.
  - **Security**: RDS Proxy supports Amazon Virtual Private Cloud (VPC) access, helping to secure your database connections within your VPC. It also supports integration with AWS Identity and Access Management (IAM) for fine-grained access control.
  - RDS Proxy is never publicly accessible, must be accessed only via VPC

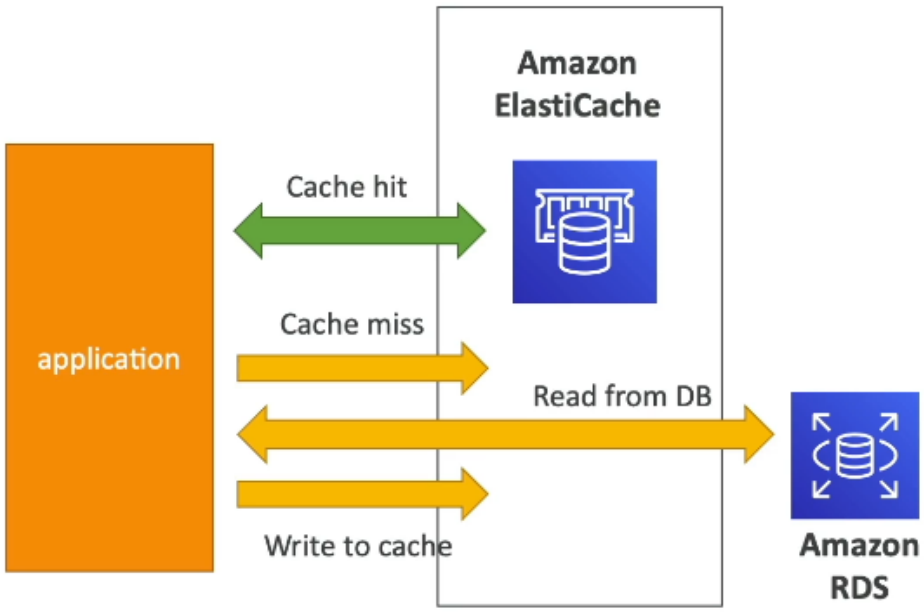
- 

- **Proxy with Lambda:**
  - Lambda functions are often short-lived and can be invoked at a high rate. Establishing new connections for each Lambda invocation can introduce overhead and impact performance.
  - RDS Proxy maintains a connection pool, reducing the overhead of creating and closing connections for each Lambda invocation. This helps improve the efficiency of your Lambda functions.

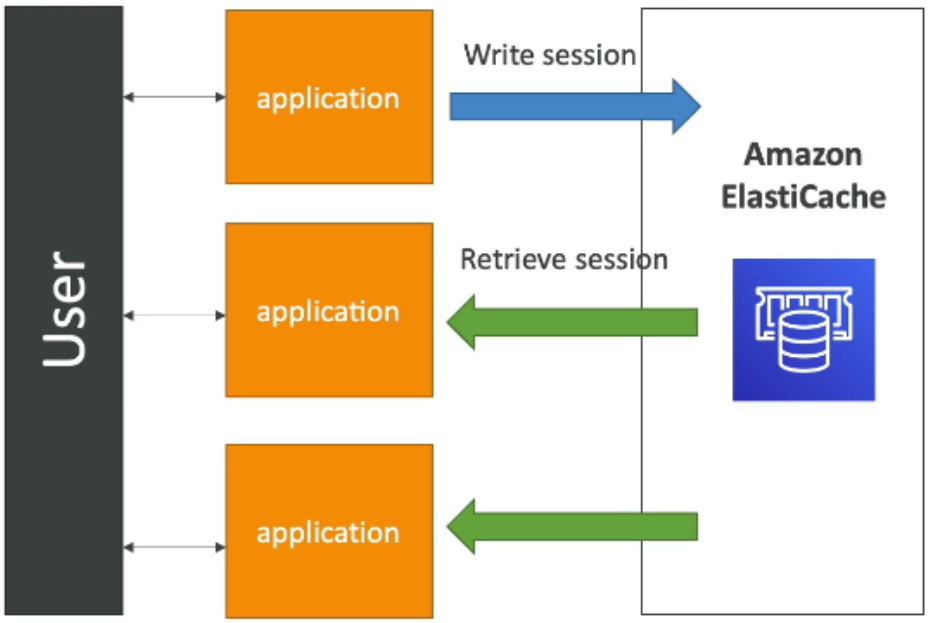
- ElastiCache

- Amazon ElastiCache is a fully managed in-memory data store service provided by AWS. It helps improve the performance and scalability of your applications by providing a caching layer that stores frequently accessed data in-memory, reducing the need to fetch data from the underlying databases.
- Provides support for two popular caching engines: Redis and Memcached
- ElastiCache is a fully managed service, meaning AWS handles tasks like provisioning, patching, scaling, and backups.

- **Use Cases:**
  - **Caching:** Improve the performance of read-heavy applications by caching frequently accessed data.



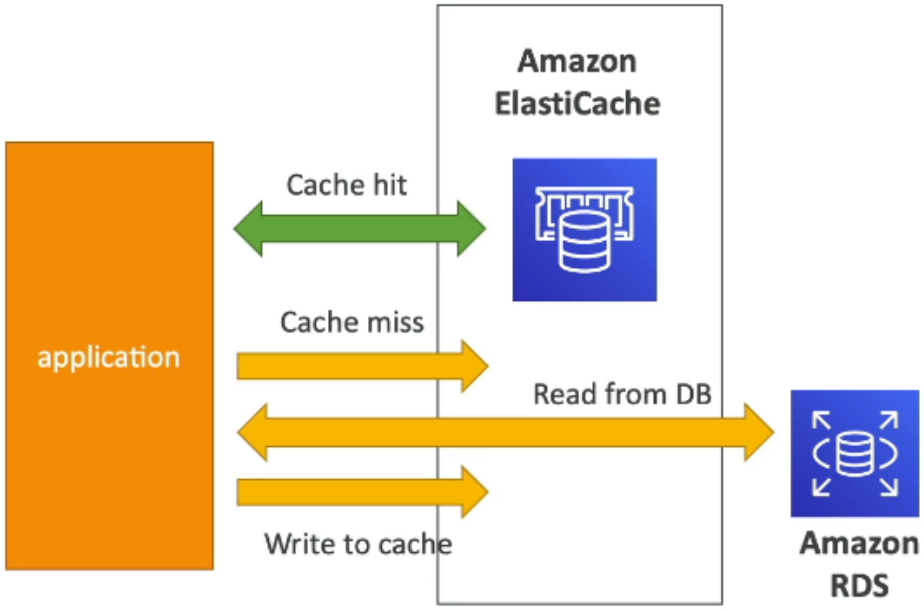
  - **Session Management:** Store user session data in the cache to reduce the need for frequent database queries.



- Redis VS Memcached
  - Redis
    - Data Persistence: Supports data persistence, suitable for durable storage.
    - Data Structures: Offers various data structures (strings, lists, sets, hashes, etc.).
    - Replication: Read replication and **Multi-AZ deployments.**
    - backups and restores
    - Auto failover
    - Security: Offers encryption, authentication, and access control.
    - Use Cases: Caching, session storage, real-time analytics, complex data processing.
  - Memcached
    - Simplicity: Simple and straightforward to use.
    - Multi-node for data partitioning (sharding)
    - Data Persistence: Does not provide built-in data persistence.
    - Replication: No replication
    - No backups and restores
    - Multi-threaded

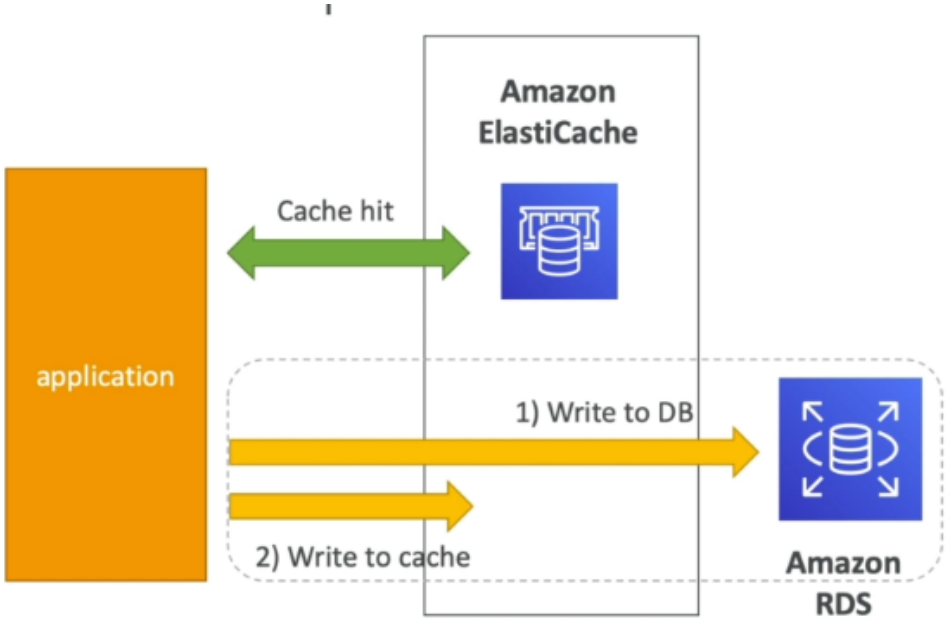
        - Use Cases: Basic caching needs, high-speed data retrieval, load reduction on backend.
        - 

- **ElastiCache Strategies**
    - **Lazy Loading/Cache-Aside/Lazy Population**



        - This means that data is loaded into the cache only when it is needed.
        - Use Case: Applications with limited cache capacity.
        - The Cons in this strategy are:
            - Cache Read miss penalty:
                - if the data is not on the cache then 3 round trips happens. one for cache miss, one for getting data from database and one for write data to cache.
                - Stale data: data can be updated in the database and outdated in the cache.

    - **Write Through**

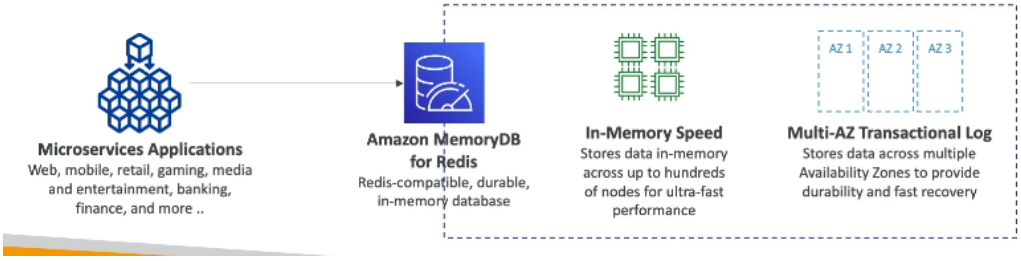


- In write through, data is written to both the cache and the underlying data store simultaneously whenever a write operation occurs. This ensures that the data in the cache is always synchronized with the data in the primary data store.
- Pros:
  - Data in cache is never stale
  - Each write in database also written in cache (Write Penalty)
- Cons
  - Increased latency for write operations, as data must be written to both the cache and the underlying store.
  - Increased write load on the cache, which might impact cache performance.
  - Cache churn: A lot of data will never be read

- **Cache Eviction**
  - Cache Eviction refers to the process of removing or replacing items from the cache to make room for new or more relevant items. Since cache space is often limited, eviction mechanisms are used to decide which items to remove when the cache becomes full.
  - Different eviction policies determine how items are selected for removal:

        - **Least Recently Used (LRU):** Evicts the least recently accessed items first. Items that haven't been accessed recently are more likely to be evicted.

    - **Time-To-Live**
        - TTL is a setting associated with cached items that specifies the amount of time they should remain in the cache before being automatically invalidated and removed. It helps ensure that cached data remains relevant and up-to-date. When a cache entry's TTL expires, the item is considered stale and is removed from the cache during the next access attempt.
        - TTL is often specified in seconds, indicating the duration a cached item remains valid.
        - Shorter TTL values result in more frequent cache updates and may reduce the risk of serving stale data.
        - Longer TTL values can reduce cache churn and improve cache efficiency, but they might lead to serving slightly outdated data.

- Amazon MemoryDB for Redis
    - Ultra-fast performance over 160 million request/second
    - Multi AZ transaction
    - scale seamlessly from 10 GB to 100 TB of storage
    - 

Route 53

Amazon Route 53 is a scalable and highly available Domain Name System (DNS) web service offered by Amazon Web Services (AWS). It helps route traffic on the internet to various AWS resources like EC2 instances, S3 buckets, Load Balancers, and more, as well as to external resources outside of AWS.

- **Domain Registration:** You can register new domain names directly through Route 53, or you can transfer existing domain names from other registrars. ex (ASA.com --> 123.45.67.89)
- **Health Checks and Failover:** Route 53 allows you to perform health checks on your resources, This can be used for failover and disaster recovery scenario**s.**
  - **Http** health checks are only for public resources
  - **Health Check types:**
    - **Monitor an Endpoint**
      - check the resource by send requests from 15 health checkers.
      - you have the ability to choose health check region
      - the interval of requests is 30 sec can be 10 but cost higher.
      - support HTTP, HTTPS, TCP.
      - if > 18% of health checker return health then Route 53 consider it healthy.
      - health only if the endpoint responds with 2xx or 3xx status code.
      -

    - **Calculated Health Checks**
      - loop over health checks and combine their result
      - use OR, AND or NOT to test if all the checks return health or some or not
      - usage: perform maintenance to your website
      -

    - **Private Hosted Zones**
      - Used with the private Endpoints because the health checkers are from outside the VPC so they can access the private endpoints so the solution is to create CloudWatch inside the VPC and create CloudWatch Alarm to alarm when the resource become unhealth then create a health check that checks the alarm itself.

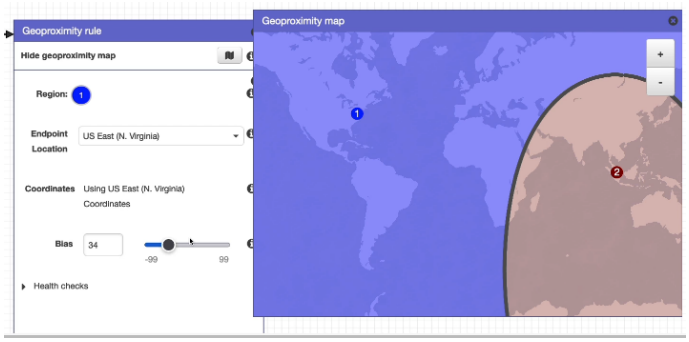
- **Records:** Amazon Route 53 allows you to create various types of DNS records to manage the mapping of domain names to IP addresses or other resources.
  - Domain/subdomain Name: example.com

- Record Type: A, AAAA, CNAME or NS
  - **A** : This record type maps a domain name to an IPv4 address. It is used to direct traffic to a specific IP address. A typical use case is to point a domain to a web server
  - **AAAA**: Similar to the A record, but it maps a domain name to an IPv6 address, enabling traffic over IPv6.
  - **CNAME:**A CNAME record is used to alias one domain name to another. It is often used to point a subdomain to another domain or to provide an easy-to-remember name for a resource that might change its IP address.
  - **NS:**
- value: 12.34.56.78
- Routing Policy: how Route 53 responds to queries
- TTL: amount of time the record cached in DNS resolver

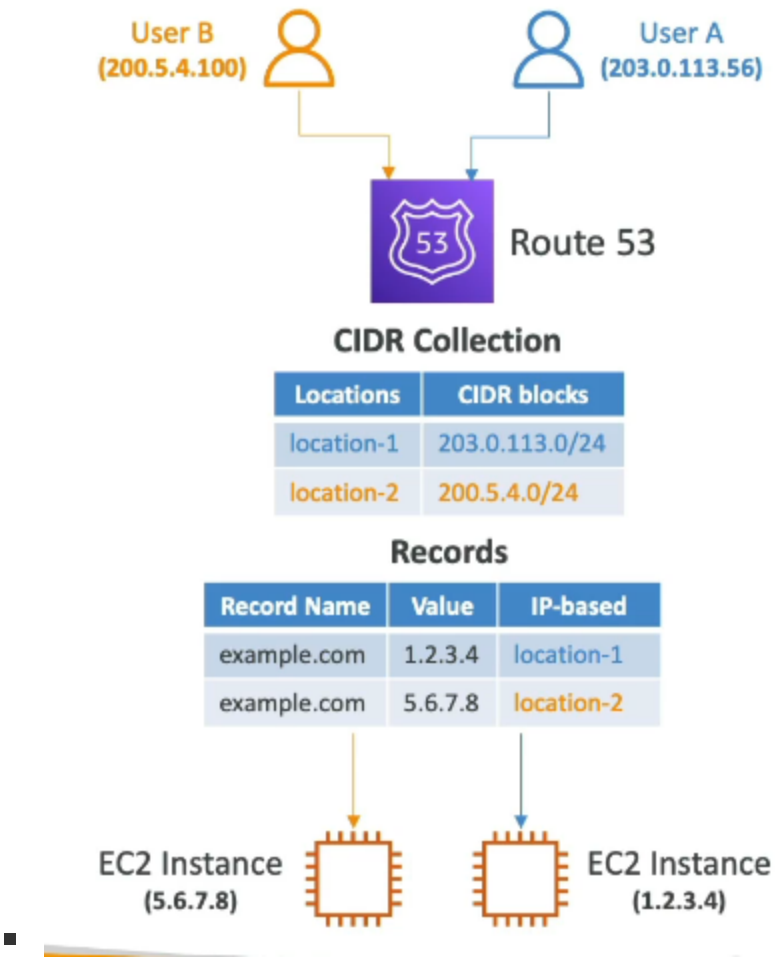
- CNAME VS Alias
  - CNAME
    - points a **NON ROOT** domain to a AWS resource hostname
    - ex: map ahmed.ASA.com to load balancer hostname alb.us-east-1a.amazonaws.com
    - **it doesn't work for ROOT domain name like ASA.com**
  - Alias
    - Free of charge
    - only point to a resource with IPV4 or IPV6
    - you can not set TTL, it sets automatically by Rout 53
    - points a **Root and non Root** domain to a AWS resource hostname
    - ex: map ahmed.ASA.com to load balancer hostname alb.us-east-1a.amazonaws.com
    - ex:map ASA.com to load balancer hostname alb.us-east-1a.amazonaws.com
    - target only (ELB, Amazon CloudFront, Amazon API Gateway, Elastic Beanstalk, S3 website, VPC interface endpoint, global accelerator, Route 53 record)
    - can not target EC2 DNS

- Routing policy
  - How Route 53 responds to DNS
    - **Simple**
      - map to single resource
      - can map to single or multiple values and random one is chosen by the client
      - **can not associate health check**
    - **Weighted**
      - control the percentage of the requests to each record based on given weight
      - records must have the same name and type .
      - used in load balancing and testing new apps.
      - assign weight 0 to a record to stop sending traffic to it.
      - if all have weight 0, then all will be returned equally.
      -
    - **Latency based**
      - redirect to the resource that has the least latency close to you.
      - helpful when latency is priority
      - latency of the user corresponding to AWS region
      - can be associated with health checker
      -
    - **Failover**
      - Create to record one as primary and other as secondary in case the first one become unhealthy the Route 53 automatically change to the secondary.
    - **Geolocation**
      - Decide if the user from a specific location (county or Continent) then respond with this DNS IP and choose Default option for other users.
    - **Geoproximity**

        - Depending on a variable called Bias, it shift the traffic from one region to another.
          if the Bias in both region is =0 then the line will be in the middle and each region will have the have of the traffic.
          this kind of record and most other records can be done using a visual feature in Route 53 called Traffic Flow:
            - Traffic Flow is a visual editor simplify the process of creating and maintaining records in large and complex configuration
- **IP-Based**
  - routing based on clients'IP address.
  - you provide list of CIDRs for clients and corresponding endpoints.
  - that optimize the performance and reduce the network cost.
- **Geoproximity**
    - Depending on a variable called Bias, it shift the traffic from one region to another.

- if the Bias in both region is =0 then the line will be in the middle and each region will have the have of the traffic. this kind of record and most other records can be done using a visual feature in Route 53 called Traffic Flow:
    - Traffic Flow is a visual editor simplify the process of creating and maintaining records in large and complex configuration
    - 

- **IP-Based**
    - routing based on clients'IP address.
    - you provide list of CIDRs for clients and corresponding endpoints.
    - that optimize the performance and reduce the network cost.



- **Multi-Value**
  - Use when routing traffic to multiple resources.
  - Respond with all mapping resources IPs but after make a health check
    and this is the difference between this record type and **Simple type**
  - **Simple** doesn't check the resources before return it and that may cause sending unhealthy resource to the client.
  - if unhealthy resource found by multi-value record health check it drop the unhealthy from the returned resources.

- Domain Registrar != DNS service
- But every Domain Registrar usually come with DNS features.

- You can buy a Domain from 3rd party and use Route 53 on AWS as DNS Service

---

VPC (Virtual private Cloud)

- Private network to deploy your resource
- One VPC per region
- VPC is partitioned to subnets (public-private)
- The default VPC contain public by default
- public subnet is a subnet that can access the internet and can be accessed from the internet
- private subnet can access the internet but cannot be accessed fro the internet
- Route Tables to define the access between subnets and access to the internet
-
- ### Internet Gateway And NAT Gateways
  - Internet Gateways
    - help our VPC instances connect with the internet
    - public subnet have a route to the internet Gateway and use it to access the internet
  - NAT Gateways
    - AWS Managed
    - NAT Instances is self-managed.
    - Allow the instances in private subnet to access the internet while remaining private and cannot be accessed from the internet.
    -
-
- **Network ACL (NACL) and Security Groups**
  - **NACL**
    - Control traffic flow from and to the **Subnet**
    - Allow all traffic by default
    - Can Allow and deny rules
    - **Subnet level**

        - Rules only include IP addresses
    - **Security Groups**
        - Control traffic flow from and to an ENI (Elastic Network Interface) / Instances
        - Can have **only allow** rules
        - Rules include IP addressed and other security groups.

- **VPC Flow Logs**
    - Capture information about IP traffic into your interfaces:
        - VPC
        - Subnet
        - Elastic Network Interface
    - Helps to monitor and troubleshoot issues

- **ALL The Important To Know For The Exam**
-

- **Important Architectures For The Exam**
    -

    - **LAMP STACK ON EC2**
        - **Linux**
        - **Apache server**
        - **MySQL**
        - **PHP**
        - Can add Redis/Memcached and EBS to store local apps

    -
-

- # S3
    - **Use Cases**
        - Backups
        - Disaster Recovery
        - Archive

        - Application Hosting
        - Media Hosting
        - Static website
        - Data Analysis
    - S3 must have globally unique name
    - Region level
    - Objects (files) have a key and the key is the full path => s3://my-bucket/file.txt
    - Max object size is 5 TB
    - if you uploading more than 5GB must use "multi-part upload"

    - **S3 Bucket Policies**
        - JSON used to put S3 policies
        - To Grant public access to the bucket
        - Force objects to be encrypted at upload

    - Examples
        - To allow public access (website visitor) => Use S3 bucket policy
        - To allow AWS user access S3 => IAM Policy
        - To allow EC2 instance access S3 bucket => EC2 instance role + IAM permissions
        - To allow Cross-Account-Access (Other AWS account) => S3 bucket policy (Allow Cross-Account)

    - **S3 Versioning**
        - You can version your files in S3 by enabling it at bucket level
        - its best practice to version your files
            - protect against unintended delete
            - easy roll back to previous version
        - files before enabling versioning will have version id = "null"
        - suspending versioning doesn't delete previous version

    - **S3 Replication**
        - Must enable versioning in both buckets
        - Cross-Region-Replication **CRR**
        - Same-Region-Replication **SRR**
        - Buckets can be in different accounts

        - Copying is Asynchronous => happens in background
        - Must give proper IAM permissions to S3

        - Use Cases
            - CRR => compliance, lower latency access, replication across accounts
            - SRR => log aggregation, testing

        - After replication only new object is replicated and if you wand the old objects then you can use **S3 Batch Replication feature**
        - Deletion with version ID are not replicated
        - But can replicate delete markers (disabled by default)
        - No chaining => if bucket 1 have replication into bucket 2 and bucket to has replication in bucket 3 then objects created in bucket 1 doesn't replicated in bucket 3

- **S3 Storage Classes**
    - ***Standard-General Purpose***
        - 99.99% Availability
        - used for frequent accessed data
        - low latency and high throughput
        - Use Cases => Data Analytics, mobile, and gaming applications
    - **Infrequent Access**
        - for data that is less frequently accessed but requires rapid access when needed
        - Lower cost than standard
        - ***Standard-Infrequent Access***
            - 99.9% availability
            - Use cases => Disaster Recovery and backups
        - ***One Zone-Infrequent Access***
            - High durability 99.99999999% data lost when AZ destroyed.
            - Availability 99.5.
            - storing secondary backups
        - **Glacier**
            - Low cost
            - pricing => storage + retrieval
            - ***Glacier Instant Retrieval***

        - Millisecond retrieval
        - Minimum storage 90 days
      - ***Glacier Flexible Retrieval***
        - Expedited(1 to 5 minutes), Standard(3 to 5 hours), Bulk(5 to 12) - free
      - ***Glacier Deep Archive --for long term storage--***
        - Standard(12 hours), Bulk(48 hours)
        - minimum storage 180 days
  - Intelligent Tiering
    - moves objects automatically between access tiers based on usage
    - no retrieval charges
      - ***Frequent access (automatic): default***
      - ***Infrequent access (automatic): objects not accessed for 30 days***
      - ***Archive Instant access (automatic): objects not accessed for 90 days***
      - ***Archive access tier(optional): objects not accessed from 90 to 700+ days***
      - ***Deep Archive access tier(optional): objects not accessed from 1800 to 700+ days***

  - Can move between classes manually or using S3 lifecycle configurations

S3 Lifecycle

Using lifecycle you can transition objects between storage classes

ex. for frequently accessed object move them to **standard IA**

for archive objects that you don't need fast access to move them to **Glacier or Glacier deep archive**

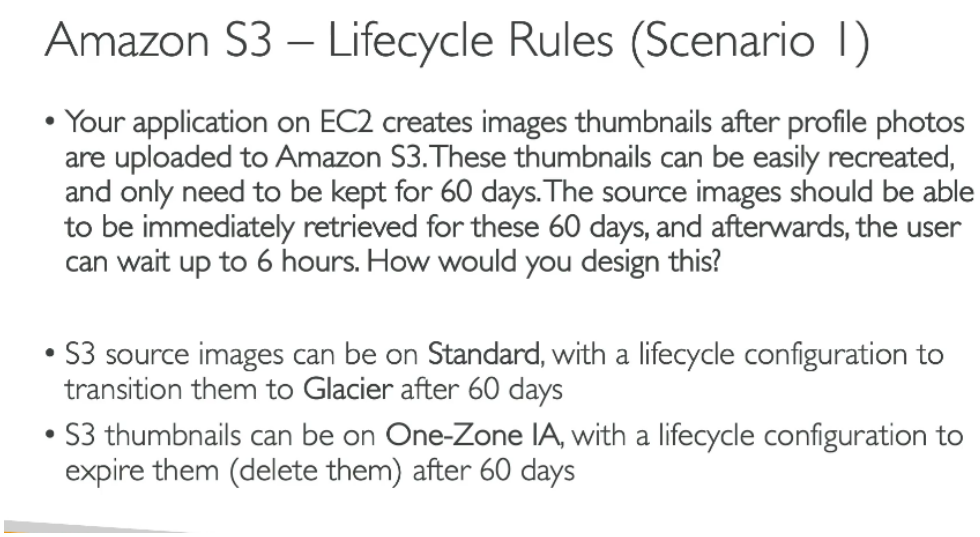
Moving objects can be done automatically using **Lifecycle Rules**

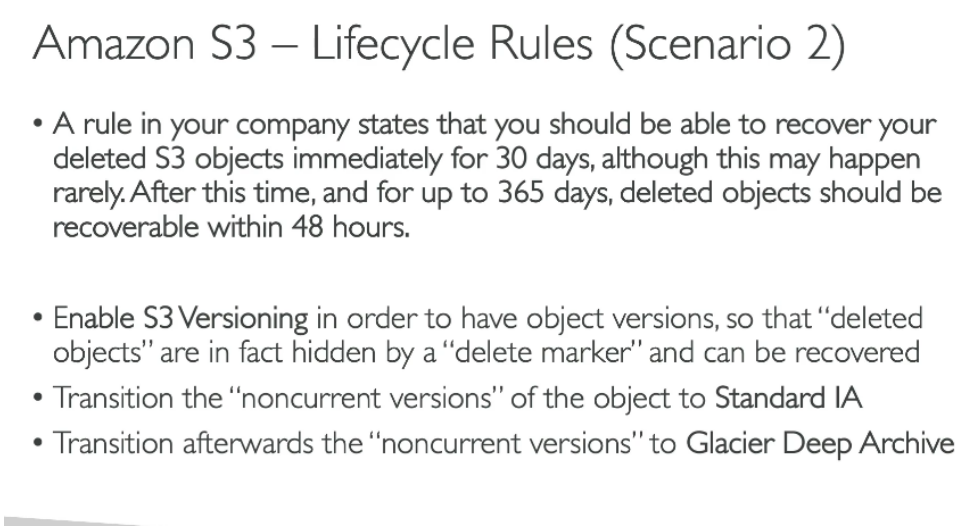
Lifecycle Rules

- Transition Action
  - Move objects to Another S3 class after certain time.
  - ex. Move to Glacier for archiving after 6 months
- Expiration actions
  - Configure objects to delete after some time

- ex. used to delete old versions (if versioning is enabled)
  used to delete incoming Multi-Part uploads
- Rules can be created for a certain prefix
  ex. *s3://mybucket/mp3/**
- Rules can be created for a certain department tags
  ex. Department: Finance

- Scenarios





S3 Analytics

Help you to decide when to transition objects to the right storage class.

CSV Report contains analysis and recommendations.

- Recommendation for **Standard and standard IA**

- Report is updated daily
- 24 to 48 hour to start seeing data analysis

S3 Event Notifications

Create a notification if something is happened

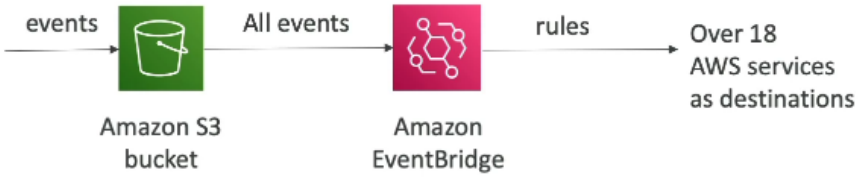
ex. S3 object => created/ removed/ restored/ replicated /etc..... or filter by name (*.jpg)

use case: create thumbnails for each image uploaded to S3

Can create as many S3 events as desired

this events can be send to another AWS service (SNS, SQS, Lambda) or to the EventBridge to be able to send messages to more services.

- **EventBridge**

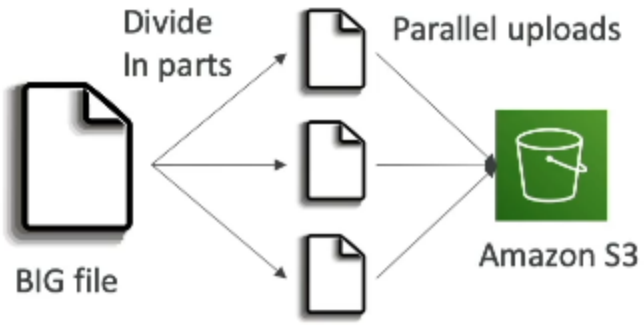


- have advanced filtering options by => name, size, metadata and more
- can send to multiple destination

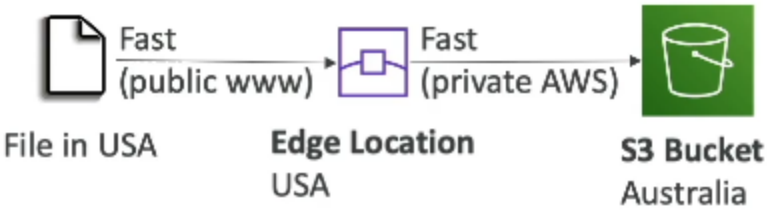
S3 Performance

- AWS automatically scales to high request rate latency 100-200 ms
- Your application can achieve at least 3,500 PUT/COPY/POST/DELETE or 5,500 GET/HEAD requests per second per prefix in a bucket.
- what is per second per prefix mean ?
  ex. S3/folder1/sub1/file ==> /folder1/sub1/
  the prefix is the right side

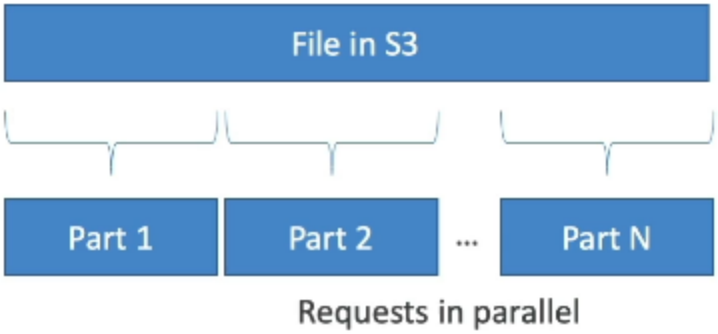
- **Increasing S3 performance**
  - Multi-Part upload
    - recommended for files > 100MB
    - Must for files > 5GB



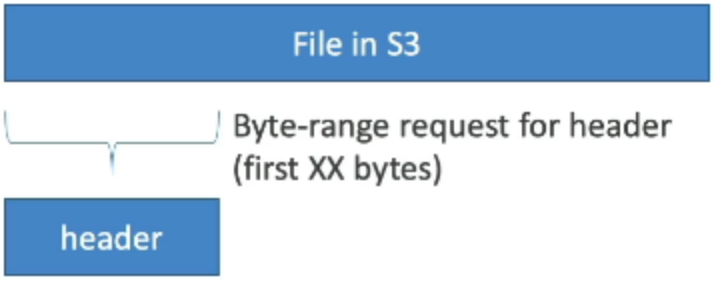
  - S3 Transfer Acceleration
    - Increasing transfer speed by transferring file to an AWS edge location which
      will transfer it to the S3 bucket using the AWS private network
    - Compatible with Multi-Part upload



- S3 Byte-Range Fetches
  - Parallelize GETs by requesting byte ranges
  - Can be used to speed up downloads



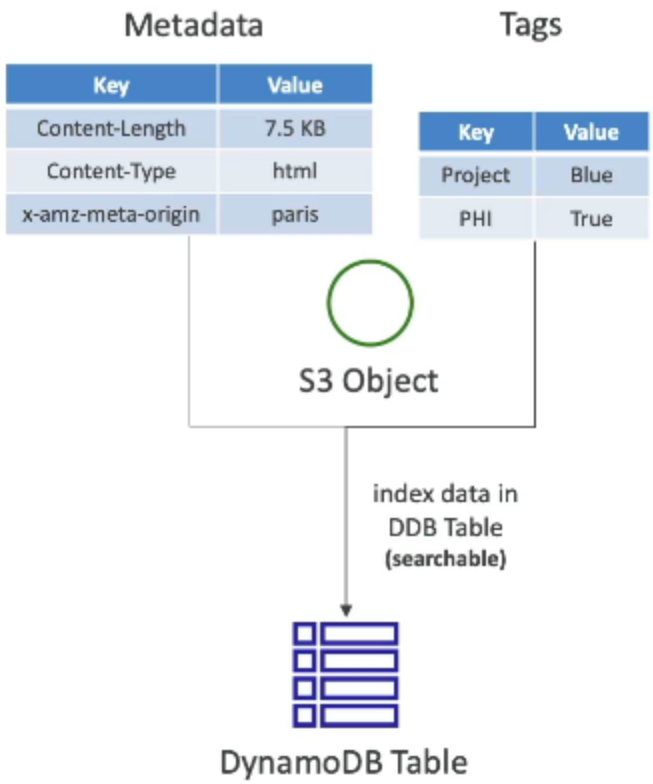
  - Can be used to retrieve only partial data



- S3 Select and Glacier Select
  - Instead retrieve all the data and then filter it you can use SQL statements to filter it first then send a small needed data.
  - Less network transfer, less CPU costs

S3 User-Defined Metadata & S3 Object Tags

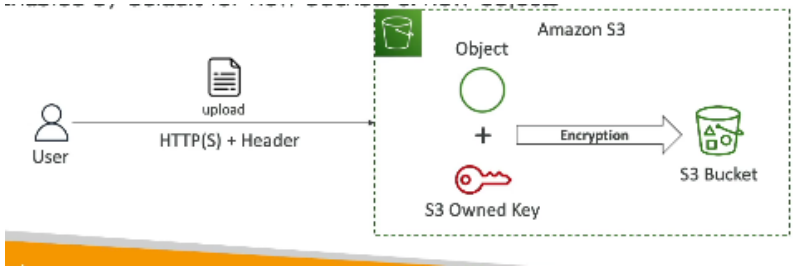
- S3 Metadata
  - When uploading an object you can also attach metadata
  - Name-Value pair
  - The name must begin with "x-amz-meta-"
  - The value stored in lowercase
  - Metadata can be retrieved when retrieving the objects
- S3 Object tags
  - Key-Value pair
  - Useful for fine grained permissions => only access specific objects with specific tags
  - useful for analysis
- You cannot search by metadata or tags but,
  Instead you make an external search Index and put the metadata and tags in this index



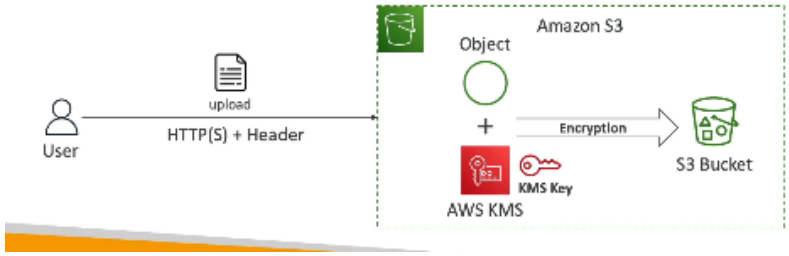
S3 Security

- **Object Encryption**

- Server-Side-Encryption SSE
  - SSE-S3
    - Encryption using keys handled, managed and owned by AWS
    - Encryption type **AES-256**
    - Must set Header **"x-amz-server-side-encryption ":"AES256"**
    - **Enable by default** for new buckets and new objects



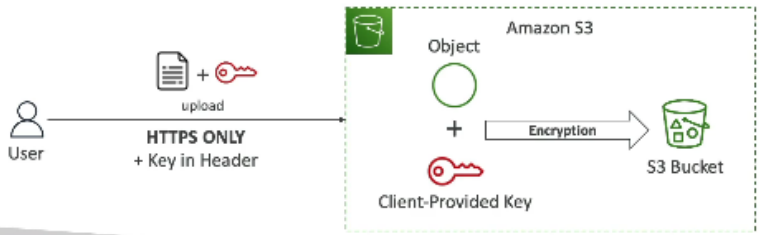
  - SSE-KMS
    - Encryption using keys handled, managed and owned by AWS KMS
    - KMS advantages: user control - audit key usage using CloudTrail
    - Must set Header **"x-amz-server-side-encryption ":"aws:kms"**



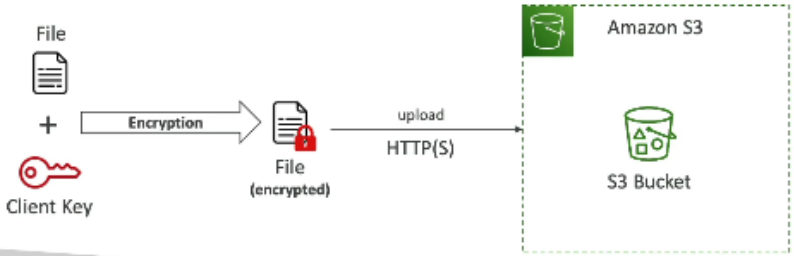
    - **KMS Limitation**
      - API calls => GenerateDataKey for upload
        => Decrypt KMS for download
      - count towards KMS quota per seconds and you can increase it.

  - SSE-C

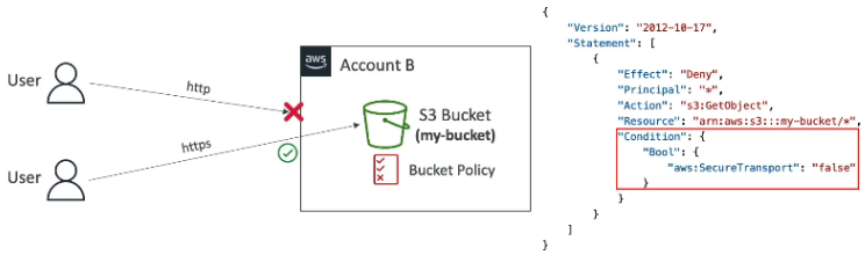
        - Encryption using keys fully managed by user outside AWS
        - AWS doesn't store the key
        - Must use HTTPS
        - Key must be provided in each HTTP request made



    - Client-Side-Encryption
        - Use client libraries in client side
        - client encrypt data before sending it
        - client decrypt data after getting it



- Encryption in transit (SSL/TLS)
    - AWS provides two endpoints HTTPS-HTTP
    - HTTPS is mandatory for SSE-C
    - U can force encryption in transit using Bucket Policy



- Also U can deny any request doesn't have a specific encryption option in the header

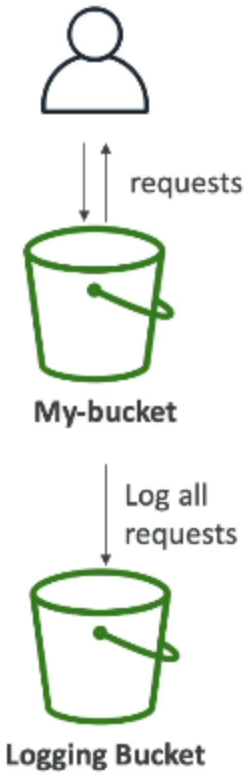
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "s3:PutObject",
            "Principal": "*",
            "Resource": "arn:aws:s3:::my-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption":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

- **S3 CORS**
  - If you request the S3 and the S3 response needs something from another S3 like an image
    then the second S3 must enable **Access-Control-Allow-Origin** for the first S3 or for all

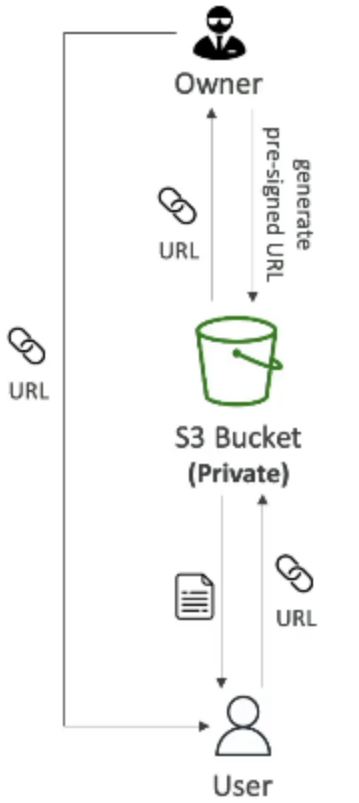
- **S3-MFA Delete**
  - will be required to permanently delete an object version or Suspend versioning on the bucket
  - Will Not be required to Enable versioning or
    List delete versions
  - Versioning must be enable to use MFA and Only the ROOT account can
    Enable/Disable it and only can do that using CLI not Consol.

- **S3 Access Logs**
  - You may want to log all access to S3 buckets and any request made to S3 from any account
    authorized or denied will be logged into **another S3**
  - The target logging bucket must be in the same AWS region

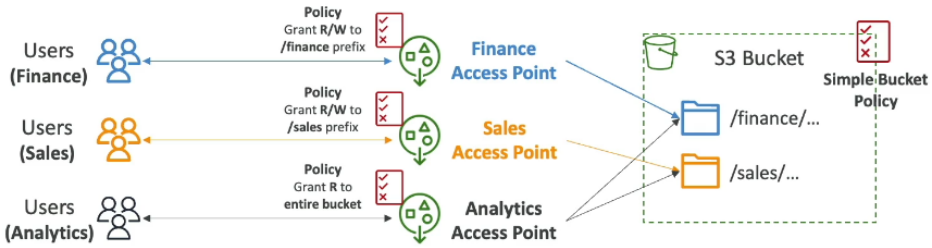


-
- Don't put the logs in the same bucket, because it will create a logging loop.

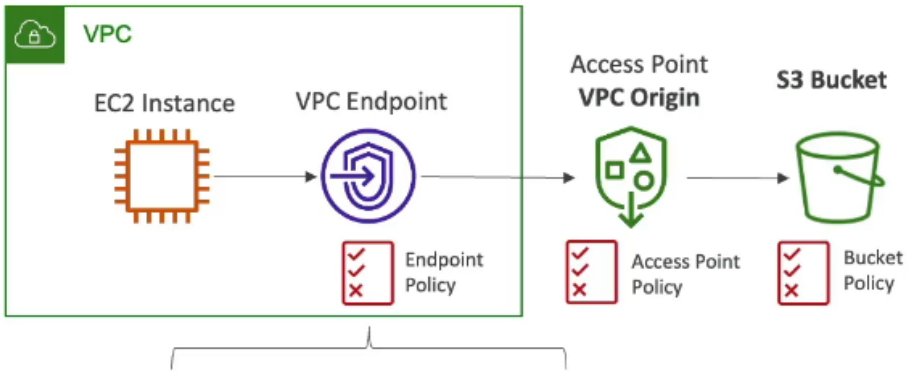
- **S3 - Pre-Signed URLS**
  - A generated URL using the S3 object used for temporary accessing S3 for certain amount of time.
  - The pr-signed URL inherit the permissions of the user for GET/PUT



- **S3 Access Points**
  - Simplify security management for S3 buckets

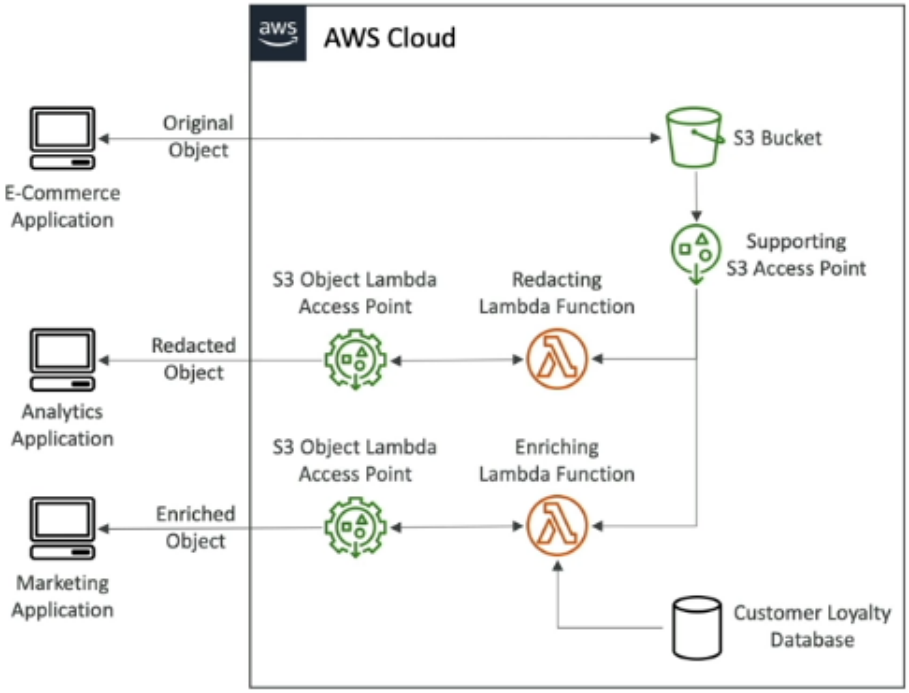


  - Each access point has => DNS name (Internet or VPC) and access point policy
  - We can define the access point to be accessible only within the VPC
  - You must create a VPC endpoint to access the access point
  - The VPC must allow access to the target bucket and access point




```
{
    "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 Object Lambda**
  - AWS Lambda used with S3 to change or modify the object before retrieving it
  - creating S3 access point to lambda function and Lambda access point to be accessed by user
  - Use Case => converting data format such as from XML to JSON.
    => watermarking an image.



- **CLI**
  - --dry--run
    - CLI command used to check if the action that i want to do is authorized or not.
    - prevent you to do the action just for know if you authorized or not.

```
[ec2-user@ip-172-31-3-136 ~]$ aws ec2 run-instances --dry-run --image-id ami-06340c8c12baa6a09 --instance-type t2.micro

An error occurred (DryRunOperation) when calling the RunInstances operation: Request would have succeeded, but DryRun flag is set.
[ec2-user@ip-172-31-3-136 ~]$
```

  - STS
    - when you use the CLI commands you may get an error message contain encoded message.
    - to decode this message you can use the command below

```
sts decode-authorization-message --encoded--message (put the message here)
```

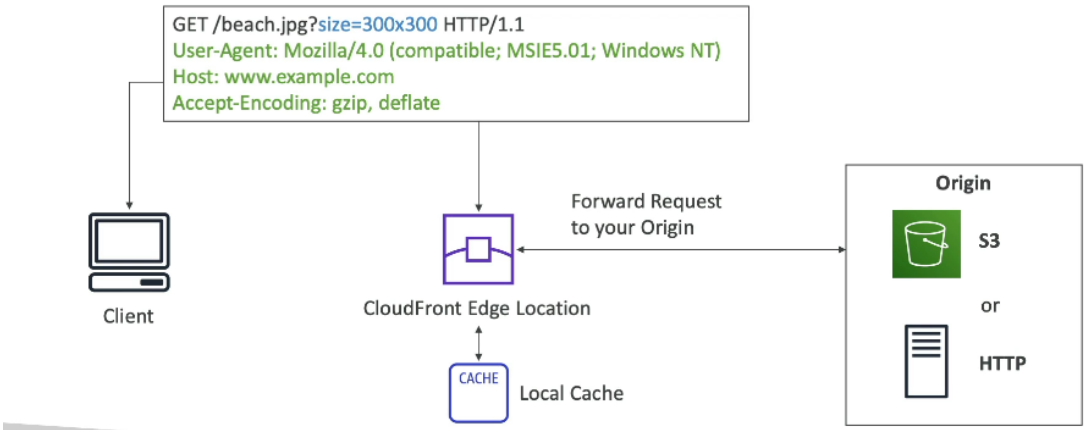
    - but these command need to be added to IAM policies to can use it.
    - this command return a JSON object contain the message details.

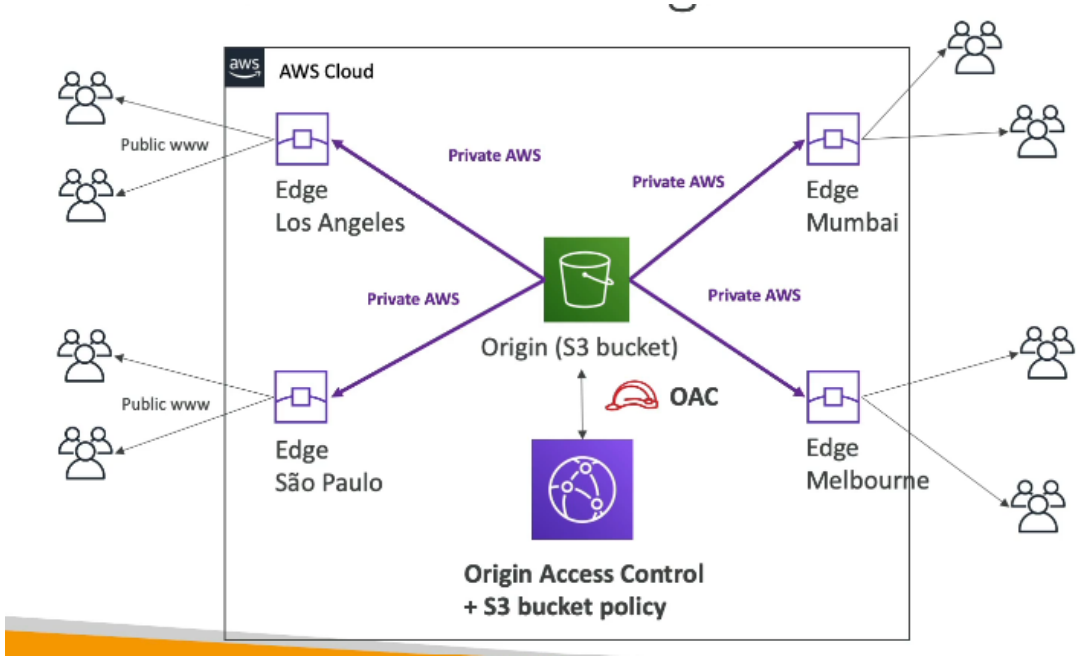
```
{
    "DecodedMessage": "{\"allowed\":false,\"explicitDeny\":false,\"matchedStatements\":{\"items\":[]},\"failures\":{\"items\":[]},\"cont
ext\":{\"principal\":{\"id\":\"AROAJREKPBRLWI6HE52M2:i-05adcce6933809eda\",\"arn\":\"arn:aws:sts::387124123361:assumed-role/MyFirstEC2Ro
le/i-05adcce6933809eda\"},\"action\":\"ec2:RunInstances\",\"resource\":\"arn:aws:ec2:eu-west-3:387124123361:instance/*\",\"conditions\":
{\"items\":[{\"key\":\"ec2:InstanceMarketType\",\"values\":{\"items\":[{\"value\":\"on-demand\"}]}},{\"key\":\"aws:Resource\",\"values\"
:{\"items\":[{\"value\":\"instance/*\"}]}},{\"key\":\"aws:Account\",\"values\":{\"items\":[{\"value\":\"387124123361\"}]}},{\"key\":\"ec
2:AvailabilityZone\",\"values\":{\"items\":[{\"value\":\"eu-west-3c\"}]}},{\"key\":\"ec2:ebsOptimized\",\"values\":{\"items\":[{\"value\
":\"false\"}]}},{\"key\":\"ec2:IsLaunchTemplateResource\",\"values\":{\"items\":[{\"value\":\"false\"}]}},{\"key\":\"ec2:InstanceType\",
\"values\":{\"items\":[{\"value\":\"t2.micro\"}]}},{\"key\":\"ec2:RootDeviceType\",\"values\":{\"items\":[{\"value\":\"ebs\"}]}},{\"key\
":\"aws:Region\",\"values\":{\"items\":[{\"value\":\"eu-west-3\"}]}},{\"key\":\"aws:Service\",\"values\":{\"items\":[{\"value\":\"ec2\"}
]}},{\"key\":\"ec2:InstanceID\",\"values\":{\"items\":[{\"value\":\"*\"}]}},{\"key\":\"aws:Type\",\"values\":{\"items\":[{\"value\":\"in
stance\"}]}},{\"key\":\"ec2:Tenancy\",\"values\":{\"items\":[{\"value\":\"default\"}]}},{\"key\":\"ec2:Region\",\"values\":{\"items\":[{
\"value\":\"eu-west-3\"}]}},{\"key\":\"aws:ARN\",\"values\":{\"items\":[{\"value\":\"arn:aws:ec2:eu-west-3:387124123361:instance/*\"}]}}
]}}}"
}
```


- Then you can use any JSON formatter to make it clear.

CloudFront (CDN)

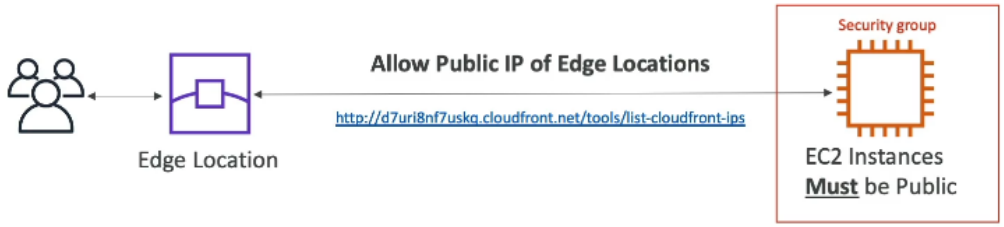
- Content Delivery Network
- Improves read performance => content is cached at thee edge
- DDoS protection because its worldwide
- OAC option => make the resource (S3) only accessible by CloudFront
- First data need is gotten from the source and cached then gotten directly from the CDN next times.
- Can be used as ingress in S3 => upload files
- Can be used with Custom Origin HTTP
  - Application Load Balancer
  - EC2 instances





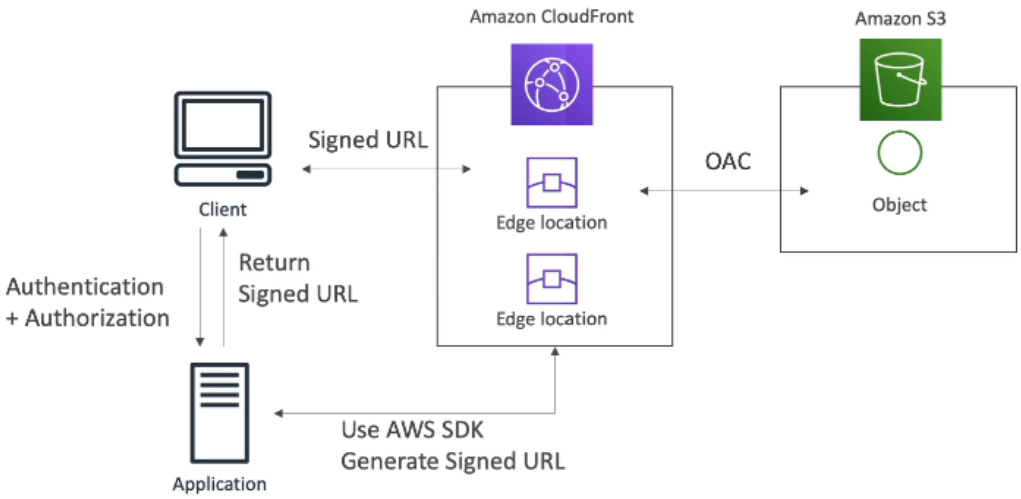
    - Common exam question:
      comparison between CloudFront and S3 Cross Region Replication.
- **Caching Policies**
    - The cache lives at each CloudFront
    - CloudFront identifies each object in the cache using the **Cache Key**
    - You can remove part of the cache using the **CreateInvalidation** API
    - **Cache Key**
        - Unique identifier for each object
        - Consist of **hostname + portion of resource URL**
        - You can add other elements to the cache key using **CloudFront Cache Policies**
        - You can include all the headers or query strings to the cache key, none of them or whitelist
    - You can control the TTL by the origin from 0 second to 1 year using **Cache-Control** header or **Expires** header
    - All HTTP headers or query strings that included in the cache key are automatically included in origin requests
    - Also using the **Origin Request Policy** you specify values that you want to include in origin request without including them in the cache key.
- **Invalidation**

    - You can invalidate all files (*) or a special path (/images/*) or specific file with its path.
- **ALP and EC2**
    - Can be used with EC2 but it must be public



    - Also it can be used with ALP and must be public.

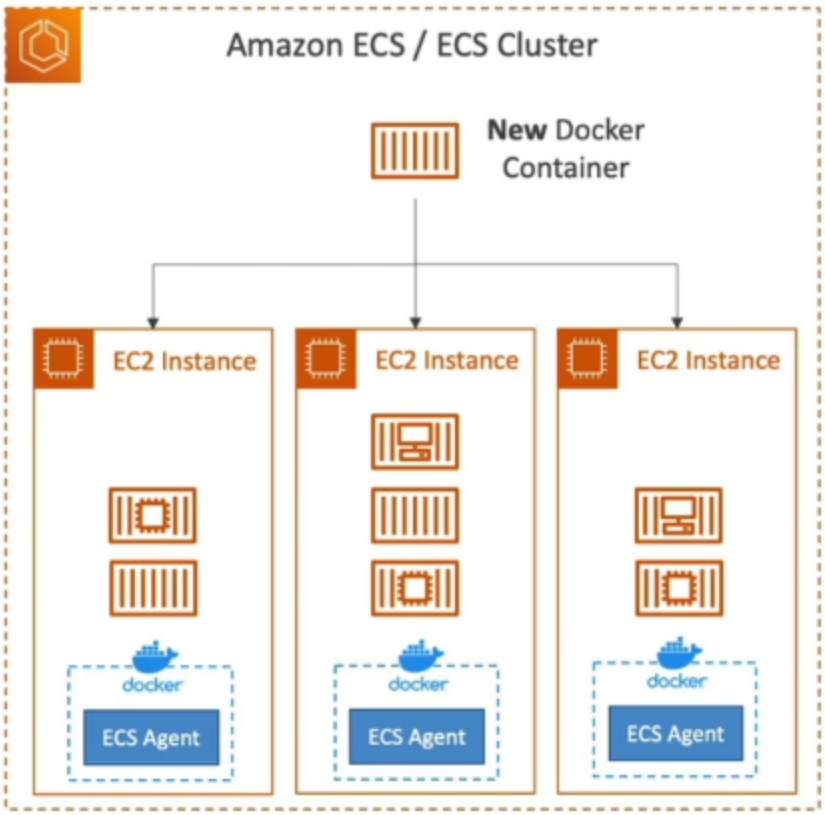
- **Geo Restriction**
    - You can restrict which country can access the distribution => **Allowlist, Blocklist**
- **CloudFront Signed URL / Signed Cookies**
    - You want to distribute paid shared content to premium users over the world
    - We can use CloudFront Signed URL / Cookie by attach a policy with:
        - URL expiration
        - IP ranges to access data if possible
        - Trusted signers (which AWS account can create signed URL)
    - How long should the URL be valid for?
        - shared content: make it short
        - private content: can be years
    - Signed URL=> one signed URL per file.
    - Signed Cookie => one signed URL per multiple files.



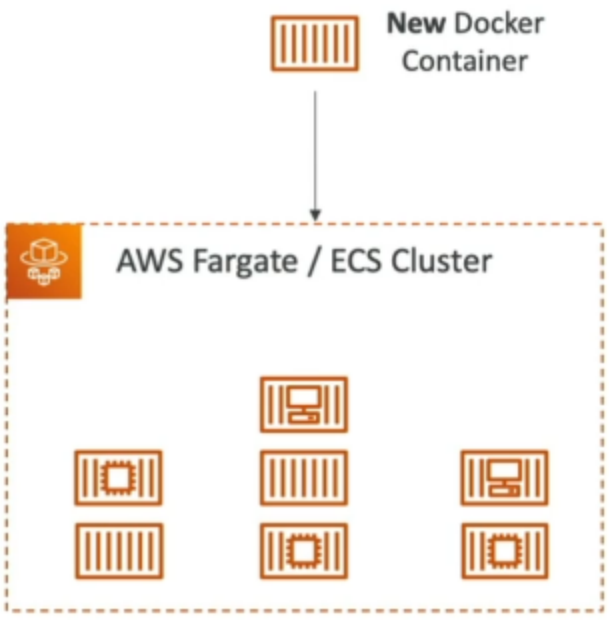
Amazon ECS

Elastic container service

- **EC2 Launch Type**
  - Must provision and maintain the infrastructure
  - Each EC2 instance must run the ECS agent and register the ECS cluster
  - AWS takes care of starting and stopping the containers

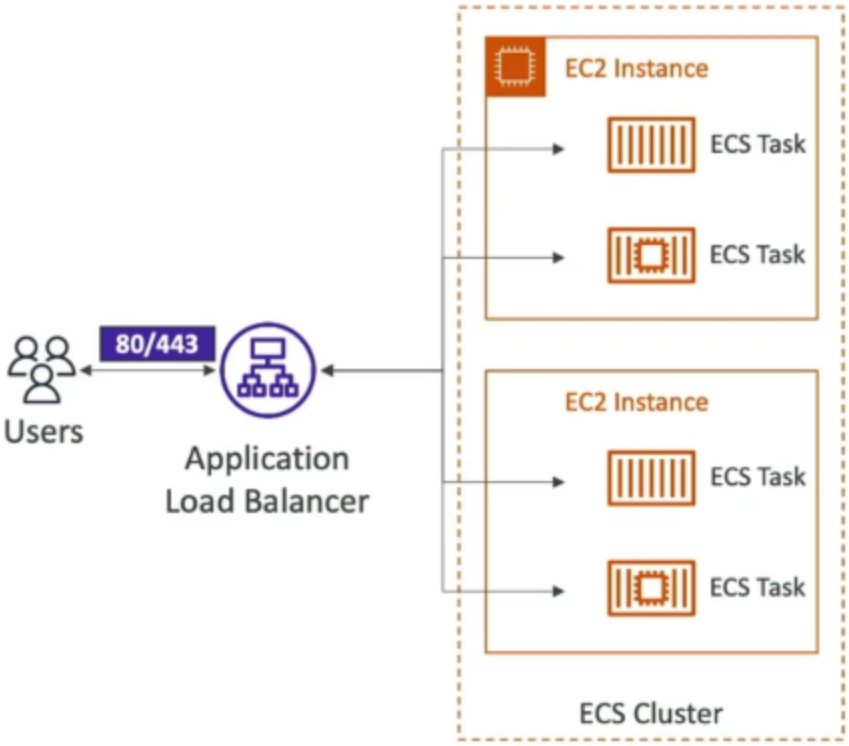


- **Fargate Launch Type**
  - You don't provision the infrastructure and no EC2 to manage
  - All is serverless
  - AWS runs ESC tasks based on CPU/RAM you need
  - To scale just increase the number of tasks



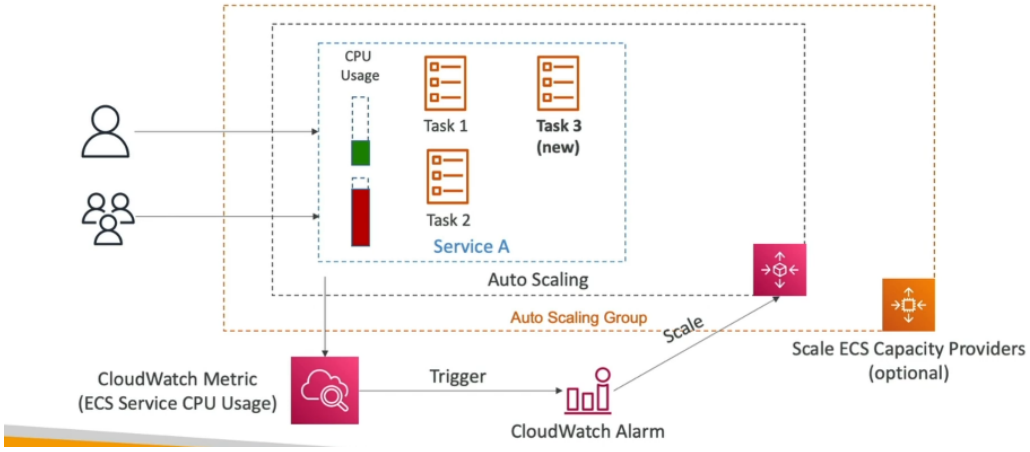
- **IAM Roles For ECS**
  - EC2 Instance Profile
    - Used by ECS agent
    - Makes API calls to ECS service
    - Send container logs to CloudWatch
    - Pull docker images from ECR
    - Reference sensitive data in Secret Manager or SSM Parameter Store
  - ECS Task Role
    - Allows each task to have a role
    - Use the roles with different ECS services

- **Load Balancer Integration**
  - ALP supported and work for most cases

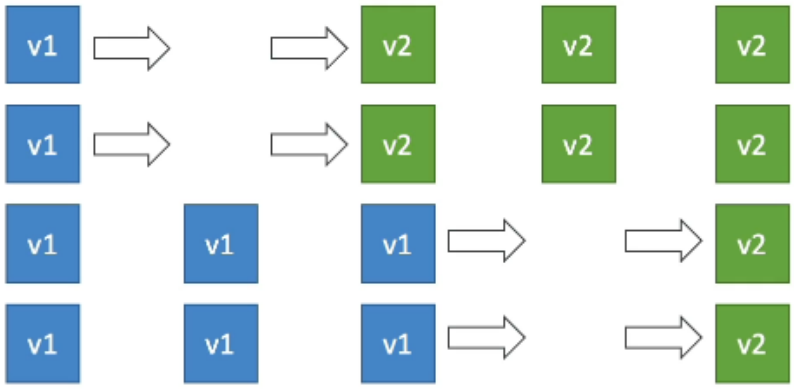


    - NLP recommended only for high throughput / high performance need or to pair with Private Link
    - ELB supported but not recommended and no Fargate support.

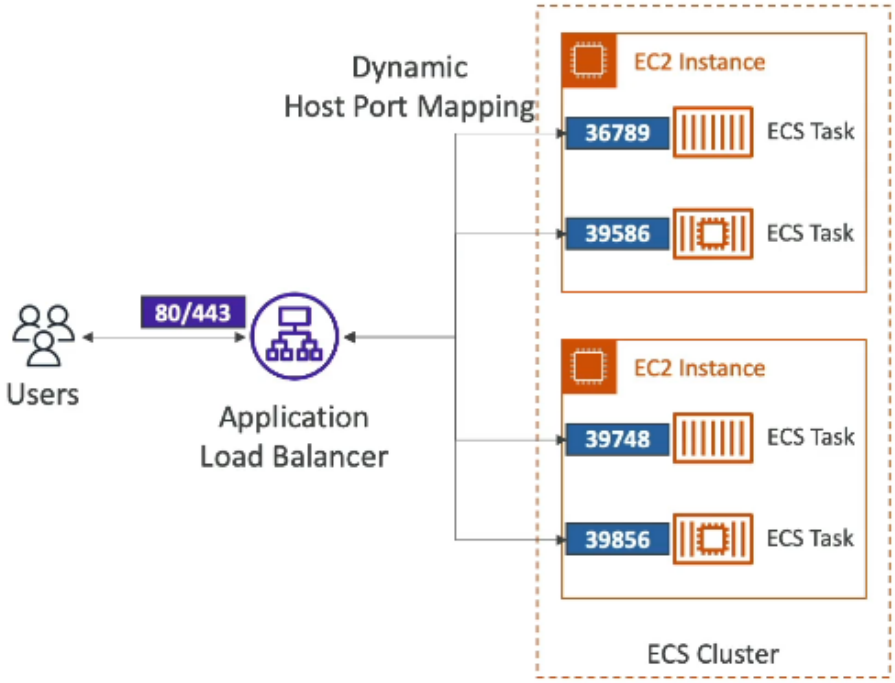
- **Data Volumes (EFS)**
    - Works for both EC2 and Fargate
    - Tasks running in any AZ will share the same data in the EFS
    - **Fargate + EFS = Pure Serverless**
    - S3 cannot be mount as filesystem
    - **ECS Scaling**
        - Service auto scaling
            - increase/decrease number of ECS tasks
            - AWS ECS auto scaling using AWS Application auto scaling
                - Average CPU
                - Average RAM
                - ALP metric
        - Target tracking
            - scale based on target value for a specific CloudWatch metric

        - step scaling
            - scale based on trigger from CloudWatch
        - Scheduled scaling
            - based on a specified date/time (predictive changes)
        - 

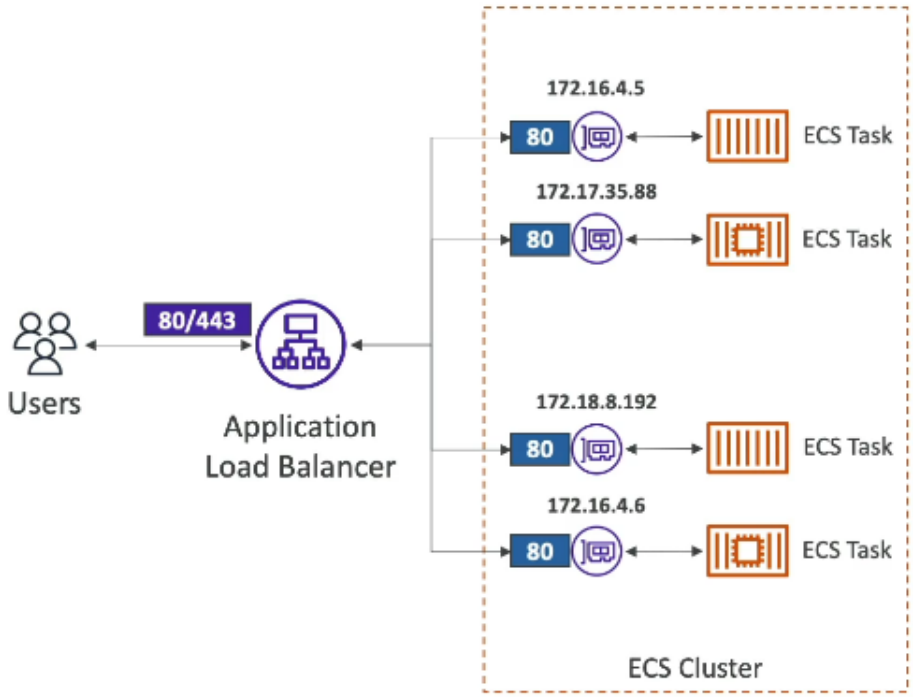
    - EC2 auto scaling
        - scaling by adding EC2 instances
        - Auto scaling group
            - scale based on CPU
            - add EC2 over time
        - ECS Cluster Capacity Provider
            - automatically provision and scale the infrastructure
            - paired with ASG
            - add EC2 when you are missing capacity
- **ECS Rolling Updates**
    - defining the percentage of tasks that must be available during the update
    - Min and Max
    - ex: Min = 50% , Max = 100% and tasks is 4 in v1
        - you can terminate 2 v1 and add 2 v2 then terminate other 2 v1 and add 2 v2



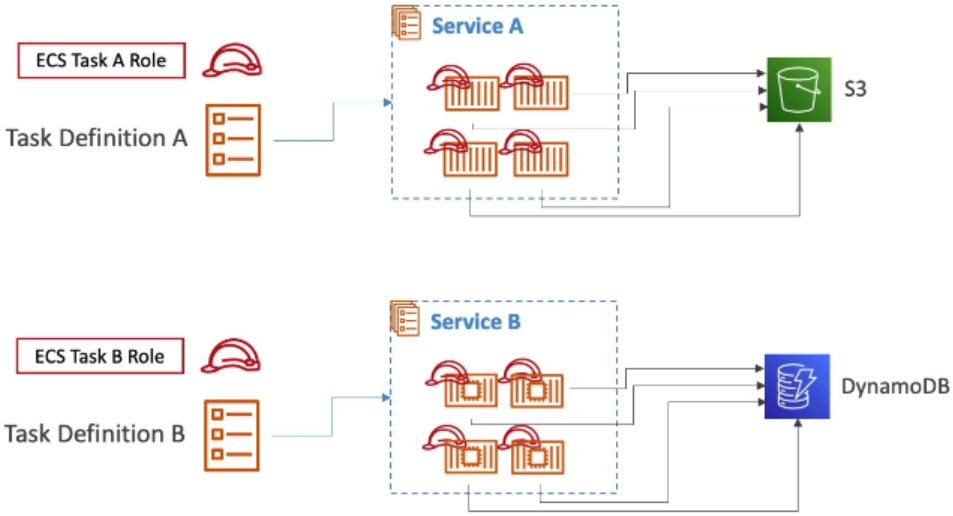
- **Task Definition**
  - Meta data telling ECS how to run docker container
    - image name
    - port binding
    - memory and CPU
    - environment variables
    - Networking information
    - IAM role
    - logging configuration
    - up to 10 containers
  - if you define only the container port then you will get a Dynamic Host Port Mapping
    and ALP will use this feature to find the containers within the EC2 but you must allow any port from the EC2.



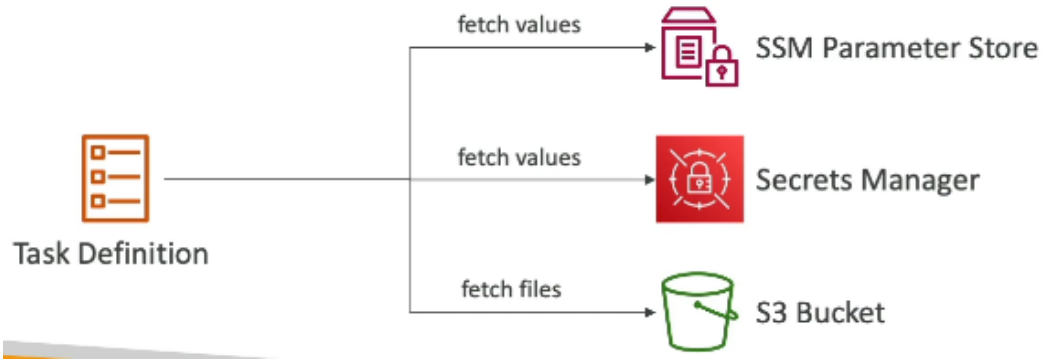
- **Fargate Load balancing**
  - In Fargate each task has a unique IP so no need to host port, the container port is enough



- **ECS IAM Role**
  - One IAM role per task definition.
  - the task service inherit the IAM role from the task definition role.

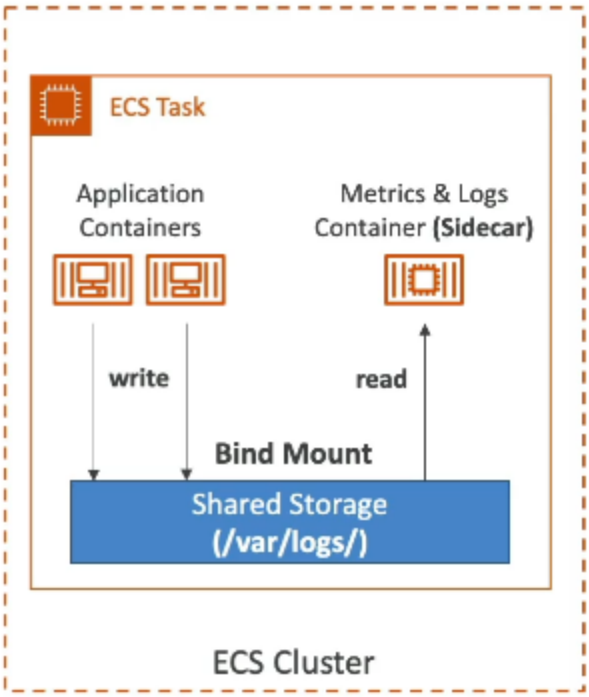


- **ECS Environment Variables**
  - Hardcoded
  - From SSM Parameter Store
  - From Secret Manager
  - From Environment File (bulk) - Amazon S3

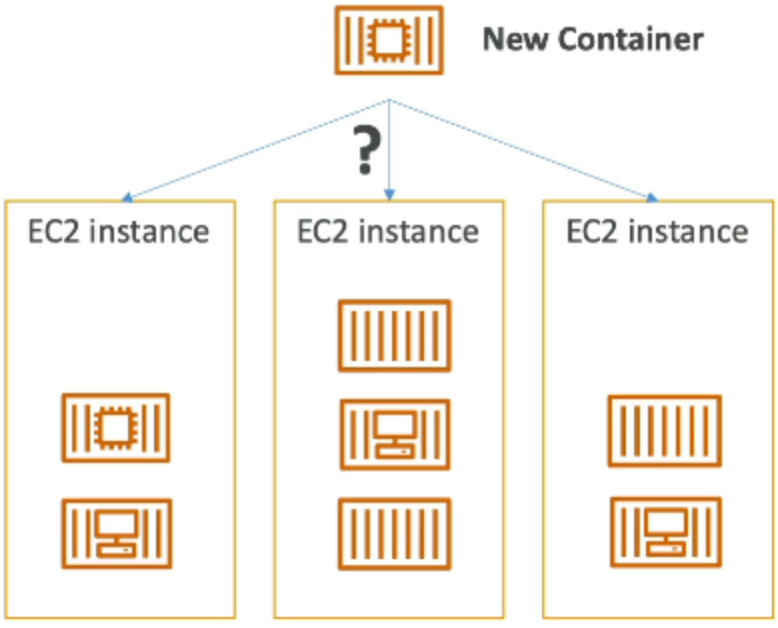


- **Data Volumes (Bind Mount)**
  - Share data between multiple containers in the same task definition
  - Works for both EC2 and Fargate
    - EC2
      - Data tied to the EC2
    - Fargate

    - Data tied to the container => 20 GB to 200 GB - 20 by default



- **ECS Task Placement**
    - You have a cluster and EC2 instance/s within this cluster and you adding a new task
      in which EC2 instance would you put this new task ??



- To determine you can define **task placement strategy** and **task placement constrain**
- This is only for ECS with EC2 **NOT** for Fargate
- **Task Placement Strategies**
  - Binpack
    - place based on the least available amount of CPU and memory
    - add to the EC2 as much as possible
    - cost saving strategy
  - Random
    - place the task randomly
  - Spread
    - place the task based on specified value
- You can mix strategies together
- **Task Placement Constraints**
  - distincitInstances => each task on different container
  - MemberOf => place task based on expression