

AWS TRAINING NOTES

AWS Certified Solution Architect – Associate Topic wise breakup

Domain	% in the Examination
1: Design Resilient Architecture	34%
2. Define Performant Architecture	24%
3. Specify Secure Application Architecture	26%
4. Design Cost optimized Architecture	10%
5. Define Operation Excellent Architecture	6%

CONTENTS

General Overview	2
Virtual Private Cloud (VPC)	4
Security Group (SG)	7
Network Access List (NACL)	7
NAT instance & NAT gateway	7
Egress Only - Internet Gateway	9
Direct Connect (DX)	9
Elastic Cloud Compute (EC2)	9
Elastic Block Store (EBS)	15
Elastic Load Balancer (ELB)	20
Auto Scaling (AS)	24
Relational Database Service	28
Simple Storage Service (S3)	35
Route 53	44
CloudFront	49
CloudTrail	53
Encryption – KMS / (HSM) Hardware Security Module	53
SNS (Simple Notification Service)	55
Beanstalk	56
AWS Service – Application Load Balancer	57
AWS Service – Network Load Balancer	58
AWS Service – ELB Comparison	60
AWS Service – Elastic Cache	63
AWS Service – API Gateway	64
AWS Service – AWS Lambda	65
AWS Service – Redshift	70
AWS Service – Kinesis	71
AWS Tags	72
AWS Service – EMR (Elastic Map Reduce)	72

AWS Service – SQS.....	73
AWS Service – DynamoDB	74
AWS Service – ECS	76
AWS Service – AWS Active Directory.....	77
AWS Service – CloudFormation	78
AWS Service – CloudWatch.....	79
AWS Service – Budget	80
AWS Service – Cost Explorer	81
AWS Service – OpsWorks.....	81
AWS Service – Storage Gateway, AWS Export/Import, VM Export/Import	82
Amazon -MQ.....	83
AWS-Step Functions	83
AWS Service – IAM (Identify and Access Management).....	84
AWS Cognito	88
AWS Organization	90

GENERAL OVERVIEW

Data center -> physical location where servers are hosted/located.

The components of Datacenter

- Network
- Security (*Both physical and data security*)
- Server
- WAN
- Internet
- Storage (*Block Storage & Object Storage*)
- File Share

Multi-Tenancy: The process of hosting multiple client data/servers on a single piece of physical hardware is call multi-tenancy. Client's data/traffic should be separate from the other client (tenant); thus, security is an important aspect of multi-tenancy. AWS archive virtualization using Citrix Xen hypervisor.

Different Types of Cloud offering

- Infrastructure as Service: Compute and storage alone.
- Platform as Service: Platform Service
- Database as a service: Database Service
- Software as a service: Complete software

Type of cloud:

- Private Cloud – on premises hosted virtualized server.
- Public Cloud – publicly hosted virtualized server.
- Hybrid Cloud – Mixed combination of Private and Public Cloud.

Benefit of Cloud Computing

- OPEX cost – no upfront cost.
- Reduce time to deploy
- Dev to Test environment made easier

- Elasticity – On Demand capacity
- Higher degree of automation can be archive.

Disadvantage of cloud computing are:

- Lack of visibility in areas that are maintained by service provider.

Top player: Cloud computing



Region / Availability Zone:

- Geographically separated region where AWS is hosting their data center are called as **AWS Region**.
- **Availability Zone**: Are the data centers within an AWS region where servers are hosted.

AWS feature:

- Multi Tenancy
- Own Virtual private cloud (can configure your own security policy / firewall / network traffic router).
- Compute
- Database
- Storage
- Monitoring & logging (audit trail)
- Domain Name Service (DNS) – Route 53
- Security & Encryption
- Other services (big data/machine learning/blockchain)

How to access AWS

- **AWS console**: web interface accessible from internet <https://aws.amazon.com>
- **AWS SDK**: Custom tailored APIs to be use with different programming platform like JAVA, php, python, node.js, .net, go, etc.
- **AWS Command Line Interfaces (CLI)**: Command line interface to access AWS features.
- **AWS API**: All AWS services can be access through AWS API (Application Programming Interface).

Root Account

- Enable multifactor authentication.
- Don't use root account for day to work, as it has all access – compromising the password of a root can lead to a disaster. AWS suggest least privileges access policy, provide minimum access to perform the assigned task. Once new access is required, same can be provided on the fly.

IAM (identity and access management)

User – 5000 max users, allowed by the IAM users

Role – when application needs to use other services on your behalf, roles are use. Temporary access is provided using rotating key.

Groups – Logical grouping of the users.

VIRTUAL PRIVATE CLOUD (VPC)

- In each region default VPC created by default.
- VPC is region specific, they can span across multiple availability zone. VPC cannot span across region.
- One subnet can be hosted in a single availability zone.
- VPC can be per department / per environment.
- Components of VPCs
 - o CIDR block of IP range can be selected from RFC1918 ranges
 - 10.0.0.0 – 10.255.255.255 (10/8) = 1,67,77,211 (very high)
 - 172.16.0.0 – 172.31.255.255 (172.16/12) = 10,48,571 (moderate)
 - 192.168.0.0 – 192.168.255.255 (19.168/16) = 65,531 (low)

Formula to calculate number of IPs is $[(2)^{32 - [\text{mask number}]} - 5 \text{ IP}]$

Each of the above mention IP range can be assigned within a private network; each address will be unique on that network but NOT outside the network.

- Through routing table, one subnet can interact with the other subnet. By default, routing table is attached to the VPC that allow routing between the connected subnet to route their request locally.
- Internet gateway – AWS managed service that connected instance in the subnet to interact with the internet ips.
- Security Group (last line of defense): Attached to EC2 instances.
- NACL (network access control list): Connected to the subnets.
- Virtual Private Gateway.
- IPV6 addressing: ALL Public address, there is nothing like private addresses like in case of IPv4 / AWS allocates the IPV6 addresses on its own.
- Implied (default) router – connecting all subnet through routing table.
- Each subnet should be associated with ONLY one route table at any given time. IF no association is defined then the subnet is associated with the default route table.
- Each of the route table has a default entry to communicate with each subnet within the VPCs. This entry cannot be deleted or altered.
- **200 route table per VPC/ with maximum of 50 Routing entry per table.**
- VPC can have any of RFC1918 IP address or any publicly routable.
- Maximum CIDR block is /16 which will have $2^{32-16} - 5 = 65,531 \text{ address}$
- Minimum CIDR block is /28 which will have $2^{32-28} - 5 = 11 \text{ address}$
- No overlapping IP addresses, it allowed in subnet attached to a VPC.
- VPC CIDR block can be expanded by adding secondary IP address. [NO need to plan the future extension, as it allows expansion].
- Limitation for adding additional CIDR block to the existing VPC
 - Common limitation of the CIDR block, as applicable
 - CIDR block must not be same or larger than any of the existing route table entry.
 - There are few restrictions on the RFC1918 uses for expansion.
- AWS reserves first 4 and last one (*total of 5 IPs are reserved by AWS per VPC*)
 - o **First address reserved as its base IP address.**
 - o **Second address is VPC router.**
 - o **Third address is reserved for DNS routing.**
 - o **Fourth one is reserved for future use.**
 - o **Last address is for broadcasting, which is not allowed.**

Example: for a range 10.0.0.0/25 = out of 32 bit – 25 bits is reserved for network and (32-25) = 7 bits can be used for hosts. Out of those available IP addresses

The IPs that can't be use are

- First IP **10.0.0.0**

- Second IP **10.0.0.1**
- Third IP **10.0.0.2**
- Fourth IP **10.0.0.3**
- Last IP $10.0.0.(2^7 - 1) = \mathbf{10.0.0.127}$
- Internet gateway: ONLY one internet gateway per VPCs.
- Any public IP are not associated to the instance, instead in the Internet gateway (IGW) will have NAT (Network Address Translation) for that IP. It will translate public IP to Private IP.
- Routing table rules are executed from top to bottom.
 - First Rule is 172.31.0.0/13 – local (any IP that are from the subnet traffic is routed locally): **This route cannot be deleted or editable.**
 - Second rule is 0.0.0.0/0 – IGW (any other IP that are not from/to the above rule are routed to the IGW (internet).
- **Elastic IP address are charged even when they are NOT USE. They need to be de-attached from the AWS account to incurring charges.**
- [IMPORTANT] Elastic IPs are totally free, as long as they are being used by an instance. However, Amazon will charge you \$0.01/hr for each EIP that you reserve and do not use. You will be charged if you ever remap an EIP more than 100 times in a month.**
- In order to delete a resource, AWS ensure there is no dependencies.
- Security Groups:
 - There is virtual firewall, that are attached to the Elastic NIC cards.
 - Any traffic coming to the security group is inbound traffic (ingress traffic), Any traffic coming out of the security group is outbound traffic (egress traffic).
 - **Up-to 5 security group can be attached to an EC2 instances.**
 - In security group is ONLY allow RULE
 - Security groups are stateful.

Default VPC	Custom VPC
Created by default for every Region	Need to create manually.
By default <ul style="list-style-type: none"> - NACL - Security Group - Route table - Default CIDR IP addresses. - Internet Gateway connected to default Route table. 	By default <ul style="list-style-type: none"> - NACL - Security Group - Route table - IP range user has to be selected. - No internet gateway connected by default.

- Creating VPC with wizard, there are four (4) options to choose from
 - VPC with single public subnet
 - VPC with public and private subnet
 - VPC with public and private subnet and hardware VPN access
 - VPC with ONLY private subnet and hardware VPN access.
- ~~VPC peering can happen only within a single region across different AWS account.~~
- **Starting Nov 2017, AWS allow VPC peering across region.** When peered the traffic across the VPCs across region goes through AWS backbone network NOT through the internet bandwidth.
- Inter region transferring of the data rate will be applicable.
- VPC is a one-to-one peering. Transitive Routing (a.k.a. edge-to-edge) routing is not allowed in VPC Peering.
- For VPC to peer successfully, the subnet CIDR block should not overlap with each other.
- VPC peering an AWS managed service, there is NO single point of failure.
- In order to have successful peering connection between two VPCs, one need to make sure the route table on the either side (both the VPCs) are altered and ensure that they have a route entry added that routes desired inbound/outbound requests to the peered VPC.
- VPC peering is an AWS managed service. There is no single point of failure. No need to create redundant.

- AWS does not allow to use NAT instance to be used through customer gateways / VPG (virtual private gateway) connection to allow access for internet access.
- Route propagation – When connected over the customer gateway and virtual private cloud, route table can update dynamically to make them aware of their counterpart's subnet routing information.
- Max of 10 VPN connection [**SOFT-LIMIT**] can be established to single VPN gateway. However, this limit can be extended by making a request to increase the limits.
- VPN CloudHub: Building on the AWS managed VPN and AWS Direct Connect options described previously, you can securely communicate from one site to another using the AWS VPN CloudHub. The AWS VPN CloudHub operates on a simple hub-and-spoke model that you can use with or without a VPC. Use this design if you have multiple branch offices and existing internet connections and would like to implement a convenient, potentially low-cost hub-and-spoke model for primary or backup connectivity between these remote offices. AWS VPN CloudHub leverages an Amazon VPC virtual private gateway with multiple gateways, each using unique BGP autonomous system numbers (ASNs). Your gateways advertise the appropriate routes (BGP prefixes) over their VPN connections. These routing advertisements are received and readvertised to each BGP peer so that each site can send data to and receive data from the other sites. The remote network prefixes for each spoke must have unique ASNs, and the sites must not have overlapping IP ranges. Each site can also send and receive data from the VPC as if they were using a standard VPN connection.

ANY subnet that is not advertise over the BGP will not be accessible on the other side. So NOT-ALL of the subnet gets exposed when connected over AWS cloud hub.

- **VPN – keyword fast/cost-effective/high latency**
- **Direct Connection Location: DX location**
- **VPC endpoints:**
- **VPC transitive routing:**
 - o **Interface endpoints**
 - o **Gateway endpoints: ONLY for S3 and DynamoDB**
- **VPC Flow logs** – that capture IP traffic pass through your VPC. This can be created at **VPC level**, **subnet level** or at **network interface level**. The logs can be store at **cloud watch logs** or in **S3 bucket**. It can capture IP traffic going-in or going-out of your VPC.
- **DHCP Option Set:**
 - o On-premises DNS can be used for VPC environment resources.
 - o AWS Route 53 service can't be use as DNS for on premises infrastructure. As for Route 53 the traffic needs to coming from **within** or **for within** AWS network.
 - o The Dynamic Host Configuration Protocol (DHCP) provides a standard for passing configuration information to hosts on a TCP/IP network. The options field of a DHCP message contains the configuration parameters. Some of those parameters are the domain name, domain name server, and the netbios-node-type.
- **VPC Traffic Monitoring:** In case of VPC traffic monitoring, mirrored copy of the traffic reaching to an ENI of an EC2 instance can be send to another EC2 instance or Network load balancer with an UDP listener. Traffic monitoring encapsulate the mirrored traffic into VXLAN header. The mirrored source and the destination appliances can be in the same VPC or in different VPCs.
- In case of VPC flow logs, enables to logging and storing of the network traffic logs which gives insight on allowed/denied traffic, source and destination, port number, protocol number, number of packets byte counts.
- Difference between VPC logs and VPC Traffic Monitoring

VPC Logs	VPC Traffic Monitoring
Enables to gain insight on the VPC network flow logs by storing and analyzing traffic.	VPC gives deeper insight as it allows to analyzed the actual traffic including the payload.
User for trouble shooting	Use to revers engineering a network attack by analyzing the actual network traffic.

SECURITY GROUP (SG)

- Security group can be a source of a security group or can be destination of a security group.
- Security group are regional.
- Security group are implicitly deny.
- Default rule of the security group, all traffic from instance which has the same security group attached. As it has allow-all rule with inbound destination as security group of itself.
- Security groups are tied up to VPCs, it can't be shared.
- Any change in the security group is affected immediately.
- Instance guard by security group / subnet guard by NACL.

Default Security Group in Default VPC	Default Security Group in custom VPC
<ul style="list-style-type: none">- All inbound traffic originated from the security group are allowed.- All outbound traffic is allowed.	<ul style="list-style-type: none">- All inbound traffic is denying.- All outbound traffic is allowed.

NETWORK ACCESS LIST (NACL)

Security Group	NACL
Operate at instance level	Operate at subnet level
Stateful	Stateless
ONLY allow rules	Allow & Deny both rules can be configured
Evaluate all the rules first before deciding on if the network traffic is allowed or denied.	Each rule has a number – rules are evaluated from smallest to the highest.
Need to attach to instances.	Automatically attached to the subnet.
Last line of defense.	First line of defense.
	Explicit deny rules added to the NACL
Every time an instance needs to be communicated this comes into picture.	ONLY comes into picture when instances need to be communicated outside of the subnet.

NACL can block certain range of IP addresses from getting into your instances, as it has denied rules same cannot be done by security groups as they don't have deny rules. Also, it's advisable to block the IP ranges at first line of defense.

Note: Router will translate instance private IP addresses into public IP address or elastic IP addresses. Within a same VPC, route table will have default entry that will all communication within the instances of the VPC.

NOTE: NACL or security group can't operate on FQDN (Fully Qualified Domain Name), one can ONLY use a CIDR block or another security group name (in case of security group) for applying rules.

NAT INSTANCE & NAT GATEWAY

Public Subnet instances that need to be connected to internet (outbound connectivity) should be pass their traffic through a NAT instance / NAT gateway. Each NAT gateway maintain a NAT (Network Address Translation) and a PAT (Port Address Translation) that allow single PUBLIC IP to send back the reply to desired connected instance.

The NAT instance should be place at the public subnet for it communicate to the internet gateway.

To protect the NAT instances security group will be attached. A NAT instance security group must allow (security group cannot be associate to a NAT gateways)

- Traffic instance security group or the private subnet's security group as a source on port 80(HTTP) and 443(HTTPS).
- Traffic Outbound to 0.0.0.0/0 (internet) on Ports 80 and 443.
- Traffic inbound from the customer's own network on port 22(SSH) to administer the NAT instance.

For NAT instances Source-Destination check needs to be disable, by default this is enable. By default, all instance ONLY allows those traffic that are originated by their IP OR sends to their IP, but in case of NAT this is different as NAT instance acts a proxy, source-destination check needs to be disable.

NAT gateway does not support IPv6 – only egress ONLY gateway supports IPv6 protocol.

NAT Gateway	NAT instance
AWS managed service – high available within each availability zone. Note: To ensure high availability host NAT gateway on each availability zone.	User needs to develop failover mechanism for failover to ensure high availability.
Speed up to 45 Gbps.	Based on the bandwidth of the instance type used for the NAT instance.
Managed by AWS	User needs to update software, maintain patches.
Optimal for handling NAT traffic	Generic AMI, configure to handle NAT traffic
Charged based on the number of NAT gateways, duration of usages, and the data transferred through the NAT gateway.	Charges based on number of instances use, duration of the usages, type of the instances & size of the instances.
Uniform offering – user need not to decide on the - type / size/ bandwidth.	User need to decide on the type of the instance/size of the instance / bandwidth capability of the instance.
During creation need to associated with the elastic IP addresses.	User can choose public IP or elastic IP address, during the instance creation time or at any later phases.
Cannot associate security group to the NAT gateway	Can associate security group to a NAT instance to control inbound and outbound NAT traffic
Use NACL for controlling the traffic on the subnet where NAT gateway is installed	Use NACL to control the traffic on the subnet where NAT instance is installed
Use Flow Logs to capture the traffic	Use Flow Logs to capture the traffic
Port forwarding is NOT supported	User can install additional software on the NAT instance to achieve PORT forwarding.
Cannot be use as Bastion Host	Can be use as a Bastion Host
Support IP fragments for UDP protocol	Support IP fragments for UDP/TCP/IMAP protocol.
DNS filtering, egress filtering NOT possible	One can use third party products like Squid to achieve complex egress filtering for compliance need.

There are three type of IP address

Private IP address	Public IP address	Elastic IP address
Non publicly routable IP address.	These are AWS allocated publicly routable IP addresses.	These are static publicly routable IP addresses. They need to be allocated to an AWS account, up-to 5 elastic IP addresses can be allocated. When elastic IP address are in use, they are free, but if they are allocated and not in use, they are chargeable.

EGRESS ONLY - INTERNET GATEWAY

- Egress Only – Internet gateway allows instance to go out to internet over IPv6 addresses at the same time preventing any inward connections from IPv6 addresses from the internet to the instance.
- In order to connect an Egress Only Internet gateway to VPC follow the following steps
 - o Create a new Egress only internet gateway and attach it to the desired VPC.
 - o Add a new route in the route table of the private subnet to route all IPv6 or desired range of IPv6 addresses to the Egress Only Internet gateway. (*Egress only – Internet Gateway is stateful in nature, it forwards the traffic from any instance within a subnet OR from any aws service to the internet, and sends the response back to the instance.*)

Note: all IPv6 address are globally unique and thus public in nature. In order to stop instance with IPv6 address to received inbound traffic one can use Egress Only – internet gateway.

DIRECT CONNECT (DX)

- Direct Connect is a NON internet based and provide high speed, low latency, and higher performance then internet VPN.
- Direct connect required virtual interfaces to connect to AWS services, there are two type of virtual interfaces.
 - o **Private Virtual interfaces:** Private virtual interfaces connect to VPC
 - o **Public Virtual Interfaces:** Public virtual interfaces connect to AWS public services like S3, SQS, Glacier, SNS etc.
- For direct connect to connect to other AWS accounts, one need to connect using hosted virtual Interface.
- IF (Virtual Interface) is basically an 802.1Q VLAN mapped from the customer router to Direct Connect router.
- Layer 2 connection cannot be established over Direct Connection.
- You cannot use NAT instance hosted on the VPC to route datacenter traffic over internet.
- For establishing DX-Connect, one need to enable route propagation in AWS Virtual Gateway, then ensure that all route table have entry to customer on- premises environment.
- Fault tolerant
 - o **For Direct-Connection:** Establish a backup hardware VPN connection to connect VPC OR established a backup direct connection on a different colocation facility using different route to connect to the AWS VPC.
 - o **For IPsec VPN connection:** Deploy secondary VPN connection as failure.
- For seamless migration of the VPN connection to Direct Connection to avail higher bandwidth connectivity: Create a new DX-Connect, change the route tables add a route to route VPC traffic to DX-Connection.

VPN Connection	Direct (DX) -Connection
VPN Connection used IPsec encrypted network connection between customer intranet and the Amazon VPC.	Direct Connection does not involved internet, instead it established dedicated connection between customer intranet and Amazon VPC.
It's a low-cost alternative, to meet low to modest bandwidth requirements. It can be configured quickly (within minutes).	Expensive than using VPN connection. Typically use for high bandwidth requirements of 1Gbps to 10 Gbps. It takes longer time (depending on the IPS) to established DX-Connection.

ELASTIC CLOUD COMPUTE (EC2)

- EC2 purchase option type = On-Demand/Spot Instance / Reserved Instance
- EC2 tenancy type = Shared / Dedicated host
- 99.95 % up-time means ~ 22min per month downtime.
- Two types of BLOCK store
 - o **Elastic Block Store (EBS)**

- Can be use as root volume
- Persistence store
- Network attached virtual drive
- **Instance Store**
 - Can be use as root volume
 - NON-Persistence store
 - Hard drive at the virtual host where the instance is running.
 - Max up-to 10 GB can be allocated per block store device

Types of Instances:

General Purpose	Compute Optimized	Memory Optimized	Graphic Compute Instance.	Storage Optimized
Balance CPU & memory.	More CPU then Memory.	More memory then computes.	Graphic optimized	Very High I/O, low latency
Suitable for most of the purpose.	Compute & High-performance compute optimized.	Memory intensive app, DB and caching	Hugh performance and parallel computing	I/O Intensive App, data warehousing, Hadoop
Example: T1/M3/M4	Example: C1/C2	Example: R3/R4	Example: G2	Example: I2/D2

Types of Elastic Block Storage (EBS)

Solid State Drive		Hard Disk Drive		
General Purpose – SSD (gp2)	Provision IOPS (io1)	Throughput Optimized HDD (st1) Cannot be use as root volume.	Cold HDD (sc1) Cannot be use as root volume.	Magnetic EBS (HDD) (standard)
<ul style="list-style-type: none"> Common workload System boot volume Transactional workload, Virtual desktop Low latency interactive app, Small databases, DEV/Test environments, 	<ul style="list-style-type: none"> Mission Critical application I/O intensive, SQL and No-SQL databases. 	<ul style="list-style-type: none"> Ideal for streaming workload required fast consistent throughput at lower cost. Big Data Application For DB warehouse For Log processing Cannot be use as root-volume. 	<ul style="list-style-type: none"> Ideal for throughput-oriented storage for large volume of data that is infrequently access. Ideal for scenarios where cost is a major factor. 	<ul style="list-style-type: none"> Low cost drive Use for transactional workloads where performance is not dependent on the IOPS. Use for infrequently access data. Can be used as a root volume.
Sizes 1TiB – 16 TiB	Sizes 4TiB – 16 TiB	500 GiB to 16TiB	500 GiB to 16TiB	Size 1GiB to 1TiB
Max Throughput 160 MB/s	Max Throughput 500 MB/s	Max Throughput 500 MB/s	Max Throughput 250 MB/s.	Max Throughput 100 MB/s.
Max IOPS 10,000	Max IOPS 32,000	Max IOPS 500	Max IOPS 250	Max IOPS 200

- Throughput – unit of information system can process in a given amount of time.
- IOPS – input output operation performed per second.

Note: In associate level exam, the questions will be more to test understanding of the different type and their use-cases rather than IOPS numbers. However, in case of professional exam, it's important to know the IOPS value for each individual type as there might be some scenario base questions where one may have to choose answers based on the IOPS value also.

- Block Storage Mapping – mapping data volumes (root & data) to the AMI.
- EC2 instance store mapping only shows mapped EBS volumes, when queried from AWS console. To view the instance store mapping one need to queried the instance meta-data information related to block device mapping.
- Instance configuration can be changed after the launch (**even for running instance**) – EBS volume can be added or removed to/from the instance configuration however instance store cannot be added after launch. The following three property of the EBS Root volume can be changed after instance is created
 - Volume size – increase the volume size. **Volume size cannot be decreased.**
 - Volume type – change from general purpose to provision IOPS
 - Change Delete-On-Termination flag
- Point in time snapshot can be created for the EBS volume
 - This can be done manually

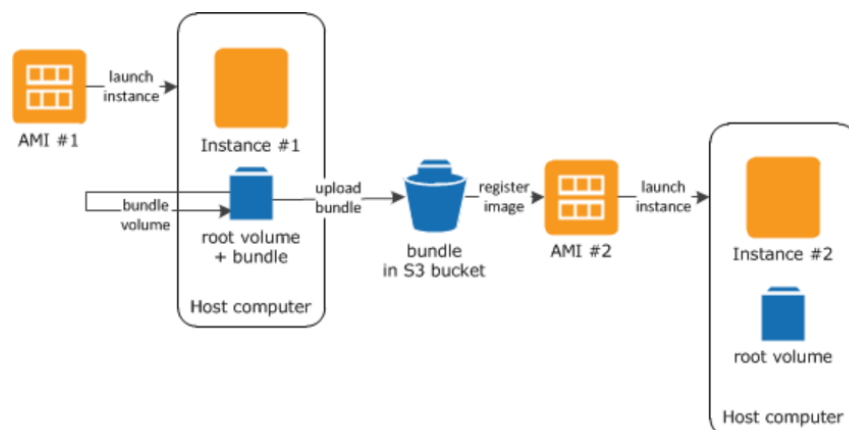
- This can be done using life cycle management -**Amazon Data Lifecycle Manager** (Amazon DLM) to automate the creation, retention, and deletion of snapshots taken to back up your Amazon EBS volumes.

Root Volume	EBS Volume
By default, delete-on-termination flag is selected.	delete-on-termination flag is NOT selected, by default. Use can select it while launching or at running state.
Cannot be encrypted while attaching.	Can be attach as encrypted.

- Default basic monitoring option available with EC2 instances are:
 - CPU utilization
 - Disk Read (bytes)
 - Disk Read Operation
 - Disk Write
 - Disk Write Operation
 - Network in
 - Network out
 - Network Packets In
 - Network Packets Out
 - Status Check Failed – ANY (count)
 - Status Check Failed – INSTANCE (count)
 - Status Check Failed – SYSTEM (count)
 - Credit Usage (count)
 - Credit Balance (count)
- Steps to create an EC2 instances with Root volume encryption
 - Create an EC2 instance based out of an AMI
 - Create an AMI out of the EC2 instance.
 - Copy the AMI to your desired region (desired region can be the same region also).
 - While creating the instance – check encrypted checkbox.
- If Instance starts – pubic IPV4 address from the pool gets allocated.
- If Instance is terminated / stop – pubic IPV4 address are release back to the pool.
- If Instance is rebooted – pubic IPV4 address is retained.
- To create an AMI out of Instance store powered EC2 instances – one need to use AWS CLI interface. AWS console doesn't give direct access to create AMI like in case of EBS backed EC2 instances. More details can be found in the below link:

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-instance-store.html>

High level process overview of the Instance Store AMI creation is :



EBS Optimized EC2 instance:

- EBS volumes are not part of the same hardware where Instance is running, they are part of AWS infrastructure connected by AWS network. The EC2 instances optimized to take full potential of the supported IOPS are called as **EBS optimized EC2 instances**.
- They are designed to work with all type of EBS types – General Purpose, Provision IOPS, Throughput optimized and Magnetic HDD
- But not all ec2 instance family are EBS optimized, only instance that are **SR-I/OV (Single Root – Input/output virtualization)**.
- In case of SR-I/OV **(Single Root – Input/output virtualization)** EC2 instances, the host's NIC (network interface card) is virtualized and share with the guest instances unlike other instances where host NIC is made available to the guest instances using the virtualization layer.
- SR-I/OV is support by R3/R4, X1, P2, C3/4, I2, M4, D2.
- **EBS enhanced EC2 instances can be enable for EBS backed EC2 instances as well as Instance Store backed EC2 instances.**
- EBS enhanced EC2 instances can be enabled for multi AZ setup.
- EBS enhanced EC2 instances are supported for
 - o Instance type should support SR-I/OV
 - o **Instance type should be created from Hardware Virtual Machine (HVM) not supported by Paravirtualization.**
 - o It's not supported for EC2 classic. Instance should be launch in VPC.
 - o There is NO extra cost in turning on the EBS optimized feature for EC2 instances (if supported).
- Placement Groups: Is a logical grouping (clustering) of the resources in a same AZ or in different AZ, with goal to reduce latency and high network throughput by determining how the instance needs to be placed on the host hardware. This can be achieved using one of the two strategy
 - o **Cluster Placement Group:** In cluster placement group setup all instances are placed in a single AZ (Availability Zone)
 - o **Spread Placement Group:** In spread placement group setup instances are placed across multiple AZ (Availability Zone)
- There is NO additional cost involved in placing instances in placement group.
- Placement Group Best practice:
 - o Ensure all instance are SR-I/OV enabled.
 - o Launch all instance together to guaranteed allocation, if the instance capacity ran out, capacity error will be thrown when launching additional instances.
 - o Try avoiding launching more than one instance type in a single placement group, to ensure capacity allocation.
 - o Can create a placement group across VPC peering connection. Both the VPCs should be in the same region.
 - o When the traffic is within instances, better to use cluster placement group.
 - o If instance is rebooted, then need to re-allocate space in underline hardware. If the requested instance type capacity ran out then capacity error will be thrown.
 - o To ensure critical application that has small number of instances, by placing then in spread placement group it ensures there is no single point of hardware failure. As the instance will be hosted in separate hardware.
 - o Within instance group, instance can communicate using private IP as well as public IP.
- EC2 instance status check:
 - o EC2 Service status check
 - o EC2 instance status check
 - o By default, EC2 instance status check will evaluated. The outcome of the EC2 instance status check is – PASS or FAIL
 - o For EBS backed volume AWS automatically restart the instance, doing so the instance get hosted in a different underlining hardware which mostly resolves the problem.
- EC2 service monitoring

- By default, EC2 service sends basic monitoring metric to cloud watch for every 5 mins (basic monitoring).
- A detailed monitoring for every 1 min can be enable, with additional cost.
- Based on EC2 service metric CloudWatch alarm can be set, depending on the cloudWatch alarm – stop/restart/terminate action can be performed.
- Different state of EC2 instance state

Pending → Running → Instance receives private as well as public IP addresses. → **Stop** AWS shutdown the instance.

- When instance is re-booted it's still considered as running state. There is additional hour added into billing, however stopping and starting an instance adds no additional hour into billing.
- Stopping a running instance, maintains instance ID & Root Volume (if its EBS backed). Instance store backed EC2 instance cannot be stop, they can be either rebooted or terminated.
- For stopped EC2 instances, they are NOT charged however for the EBS volume they will be charged.
- For stopped EC2 instance, root as well as data volume can be detached and reattached.

- **What happened when an instance is stopped?**

- Instance perform shutdown.
- Instance state change from running → stopping → stopped.
- EBS volume remain attached to the instance and it incur charges for that.
- EC2 instance doesn't incur any charges.
- Any data stored in RAM or in instance store are Lost.
- When the instance is restarted the instance may start in the same underline hardware server or a different hardware server.
- Instance IPv4 private address and IPv6 private/public address are retained.
- Instance IPv4 public address release back to AWS pool.
- Instance retains its Elastic IP addresses.

- EC2 Instance Termination

Running → Shutting down → Termination

- **What happened when an EC2 instance is Termination?**

- It will NOT incur any further cost.
- Instance Store volume data is lost.
- By default, EBS root(boot) volume is deleted and data(non-root) volume attached to the EC2 instance is retained.
- Above default behavior can be changed, by turning on the flag – **deleteOnTermination** flag during launching of the instance, during instance is running or stopped.

- Enabling **Termination Protection** on EC2 instances prevents the EC2 from accidental termination this can be enable on both EBS backed and Instance Store backed instances.
- For instances with **Termination Protection** ON, if the instances need to terminated using cloudwatch alerts then it would fail. Workaround is instead of termination try shutting down the instances for such instances.
- The flowing are the possible reasons why an instance state may move from pending to termination when launched
 - If the instance store backed AMI used for launching EC2 instances is missing or part of it is missing.
 - Limit is reached for max number of EBS volume can be attached to EC2 instance for a particular account
 - EBS snapshot is corrupted.

- Instance Metadata

- Data about the instance.

○ This can be found – <http://169.254.169.254/latest/metadata>

- User data is the script that can be pass at the time of launching. Its limited to 16 KB only, and can be change after stopping the instance. Stop instance → Instance → Action → Instance Setting → View/change user data.

- Migrating VM from/to from VMware, Microsoft Hyper V, XEN – VM Connector automate the migration of the VMware VM into AWS. More info <https://docs.aws.amazon.com/amp/latest/userguide/migrate-vms.html>
- IAM Role: IAM role can be attached to EC2 instance which enable EC2 instances to perform task on behalf of the user without any need to store IAM username/password within EC2 instances.
- Bastion Host – jump box instance which can be used to administrator other EC2 instances. AWS call bastion host for Linux instance OR remote desktop for windows instance.

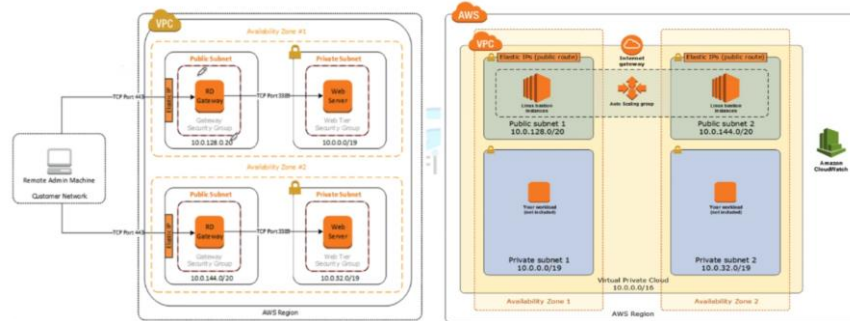


FIGURE 1 BASTION HOST HA ARCHITECTURE

- EC 2 purchasing option
 - o On demand instance [highest price]
 - o Reserved instance [paying for long term capacity for specific availability zone /region]
 - o Spot Pricing

Reserved instance – are of two types availability zone scope OR region scope

- Availability Zone scope reserved instance guaranteed capacity on that particular availability zone.
- Availability Zone scope reserved instance can be sold in marketplaces, region scope reserved instance cannot be sold in marketplaces.
- Following things can be changed for the reserved instances
 - o Change the Availability Zone for Availability Zone scope reserved instance.
 - o Availability Zone scope reserved instance to region scope reserved instances.
 - o Change the instance type in the same family
 - o One need to send request to AWS to make modification to Reserved Instances.
- Elastic Network Interface (ENI)
 - o By default, eth0 primary ENI card will be added to all EC2 instance
 - o During launch eth1 can be attached to the EC2 instance
 - o After launch, more ENI can be attached to EC2 instance based on the instance type family.
 - Attaching ENI when the instance is on **running** state is called **HOT ATTACH**.
 - Attaching ENI when the instance is on **stopped** state is called **WARM ATTACH**.
 - Attaching ENI when the instance during **launching** state is called **COLD ATTACH**.
 - o If additional ENI is attached to the EC2 instances during launch, then AWS will NOT allocate IPv4 public IP addresses to any of the ENIs, one need to attach elastic IP addresses manually. [related to scenario base questions]
 - o When instance is terminated, default ENI eth0 is terminated by default however if other ENI are attached to the instance those are not terminated by default. This behaviour can be changed as required. Even ENI create through CLI are not terminated.
 - o Each of the Elastic Network Interface can have
 - One Description.
 - One Primary IPv4 address.
 - One or Many Secondary IPv4 addresses.
 - One IPv4 public addresses assigned automatically.
 - Max Upto 5 security groups
 - One MAC address

- Source Destination check flag
- The benefit of having multiple IPv4 address are
 - It allows hosting multiple website in a single server, with each IP addresses mapped to a single SSL certificate.
 - It allows network and security appliance to be use in VPC.
 - Routing internal traffic to a standby EC2 instance on an event of primary instance failure. This can be done by remapping of the secondary IPv4 address to a secondary (standby) instances. *[When EC2 instance has secondary IPv4 private addresses, it allows reassigning of the secondary IPv4 address to new EC2 instance so that on an event of a failure, the traffic routing from elastic IP address to the secondary private IPv4 address get automatically routed to the new instance].*
- NOTE:**
 - Secondary IPv4 addresses should also be from the **same subnet of the network interface or the EC2 instance**.
 - Each private IPv4 address can be mapped to a single Elastic IP address (one-to-one relationship).
 - **Once the primary IPv4 private assigned to an elastic IP address is de-attached from the EC2 instance then Elastic IP address is also gets de-attached.**
 - One cannot de-attached eth0 (primary subnet) from the instance.
 - Private IPv4, Secondary IPv4, IPv6, Elastic IP address all belongs to the same network interface even when they are de-attached and re-attached to a different EC2 instance.
 - When attaching Network Interface to another EC2 instance – the second instance should be in the same region and should be in the same Availability Zone. Within Availability Zone, they can be in a different subnet.
 - ENI are AZ specific
- Source Destination Check flag [**question scope**]
 - By default, for all EC2 instances source destination check is enabled which restricts all traffic that leave the instance should be sourced from within.
 - For NAT instance, which acts as a proxy for the private subnet instance (instances that does not have direct connectivity to the internet) traffic that leave the instance is NOT from within the instance but from another instance. Thus, its important to disable the source destination check for NAT instances.

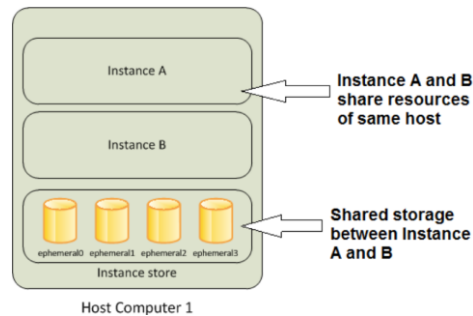
- **Spot Instance pricing:**

- If the spot instance is Launched and the AWS terminate the instance due to change in the pricing in the first hour of the billing then – THERE WILL BE NO CHARGES.
- If the users terminate the spot instance within the first hour then it will be charge till the nearest second.
- For subsequent hour (after 1st hour of operation), the spot instance will be charge for the entire time to the nearest second even if AWS terminates the instance due to the change in the stop pricing.

ELASTIC BLOCK STORE (EBS)

- There are two broader types of storage available on internet
 - **Block store** – this is primarily used for database, instance storage, for installing application etc.
 - **Object store** – for storing picture, songs, datafiles, logs files etc.
- Block store has two types
 - **Elastic Block Store (EBS)** is a network storage that is attached to any EC2 instance over AWS backbone infrastructure.
 - EBS backed EC2 instance, means the root volume of the EC2 instance is hosted on an ESB.
 - EBS backed EC2 instance can be stopped, restarted, terminated.
 - When stopped the data stored in the EBS volume is NOT lost.

- By default, EBS backed EC2 instance (*root volume*) have **deleteOnTermination flag = ON**, which deletes the EBS volume when its associated EC2 instance is terminated, however deleteOnTermination flag can be turn off if required to prevent the data on the EBS to persist when the EC2 instance is terminated.
- **Instance Store** – This is the part of the host where EC2 instance is running. Once the instance is terminated the data stored in the instance store is lost.



- Instance Store backed EC2 instance, means the root volume of the EC instance is hosted on an instance store.
- Instance Store backed EC2 instance cannot be stopped, they can ONLY be restarted or terminated.
- When EC2 instance backed by instance store is *rebooted* the data store on it will *NOT be lost*.
- If the instance is terminated the data store on the instance store volume is LOST.
- Instance Store AMI are stored in S3 instances
- EBS volume are **replicated on multiple server in a single availability zone**, to prevent single point of failure on an event of any AWS component failure.
- **Note: EBS backed EC2 instance can also have instance store which means – the root volume of the instance is on EBS and additional drive(s) which is/are hosted on instance store is attached to the EC2 instance. When such instance is stopped the data store on the instance store drives will be lost.**
- Instance Store are also called as **Ephemeral Storage**.
- Instance Store block store has higher IOPS then the its equivalent EBS block store. This is mainly because the instance store drive is on the same physical host where the instance is running whereas the EBS drives are on the NAS storage drive which is connected to the EC2 instance.
- EBS Snapshot
 - This are point-in-time copy of the EBS which can be used to recreate another image of the EBS.
 - EBS snapshot are store in S3 Bucket – however they cannot be access directly ONLY way to access them is through EC2 APIs.
 - Instance Store backed AMI snapshot are stored in user defined S3 bucket BUT EBS backed AMI snapshots are stored in S3 which cannot be access directly. ONLY through EC2-APIs these can be access.
 - Max allowed limit, 5000 EBS volume & 10000 EBS snapshot per account.
 - Characteristic of EBS Snapshot
 - They are region specific – while EBS are AZ specific snapshot are region specific. All AZ of that particular region can access the snapshot.
 - To migrate an EBS volume from one Availability Zone to another Availability Zone, create a snapshot of the volume and then create an another EBS volume in the desired AZ from that volume.
 - **[Important Exam Question]** While restoring the snapshot, the size of the new EBS volume should be same or larger than the original snapshot.
 - While EC2 instance is being accessed, EBS snapshot can be created for the **NON-ROOT volume**. However, any data cached by the OS or in memory will NOT be written into EBS snapshot. The snapshot will not be 100% consistent.
 - Phase of snapshots

Snapshot **Request Received** → Snapshot **Pending** status [till the time all the data are written into S3 bucket the status remains in pending state] → Once all the data is copied successfully the status will change to **Complete**.

- Best Practice for taking snapshot: Stop the EC2 instance then take the snapshot of the ROOT volume. For NON-ROOT volume, detached the snapshot from EC2 instance to ensure no read-write operation on the EBS volume. Once the EBS snapshot is triggered, the EBS volume can be attached back to the EC2 instance. [while the EBS snapshot is in pending state, it can be re-mount the instance].
- Snapshots occur asynchronously; the point-in-time snapshot is created immediately, but the status of the snapshot is pending until the snapshot is complete. **While it is completing, an in-progress snapshot is not affected by ongoing reads and writes to the volume.**
- Incremental Snapshot – ONLY changes that are added post EBS snapshot will be stored as incremental snapshot. [Cost effective]. Single full copy of the snapshot is need; rest can be incremental snapshot to be cost effective.
- EBS snapshot charges
 - Cost incur due for the data transfer from the EBS volume.
 - Cost of S3 storage.
 - No charges for creating snapshot.
- EBS Encryption
 - EBS encryption is supported in all EBS volume types and for all EC2 instance families.
 - Snapshot of the encrypted volume is also encrypted.
 - Following are the ways to encrypt EBS volumes (EBS encryption at rest)
 - Use encrypted EBS volume.
 - Use 3rd party tool to encrypt EBS volume
 - Use OS level encryption, using plugins and drivers.
 - Encrypt data at the application level before storing it to a disk (EBS).
 - Use Encrypted file-system on top of the EBS volume.
 - Encrypted EBS volume are to be use just like how one use non-encrypted volume. EBS Encryption is handle transparently.
 - Single EC2 instance can be attached to both encrypted and non-encrypted EBS volumes at a same time. E.g. root volume is non-encrypted, while other data volumes (non-root volumes) are encrypted.
 - There is NO direct way to change a state of the EBS volume from un-encrypted EBS volume to encrypted EBS volume, the following are the in-direct ways to archive the same.
 - Attach a new encrypted EBS volume to the EC2 instance and transfer data from un-encrypted volume to the encrypted volume. Once all data are transferred successfully de-attached the un-encrypted EBS volume.
 - Create a snapshot of the un-encrypted volume, then copy the newly EBS snapshot to another snapshot while doing check “encryption” option the resulted snapshot will be encrypted. Create an EBS volume from the new encrypted snapshot.
[EBS volume created from an encrypted snapshot is always encrypted. EBS volume created from an unencrypted snapshot is always unencrypted.]
 - Root volume encryption: There is no direct way to change the encryption state of an EBS volume. However, AWS offers the are following in-direct way to encrypt the root volume.
 - Create an EC2 instance from un-encrypted volume attached to it.
 - Create an AMI from the above created EC2 instances.
 - Copy the AMI to another location, while doing so check the encrypted option.
 - The newly copied AMI will be encrypted.
 - Create a new EC2 instance from the encrypted AMI, the new instance will have the root volume as encrypted.

While creating the snapshot of a root volume, make sure EC2 instance is stop.

- **Encryption Key:**

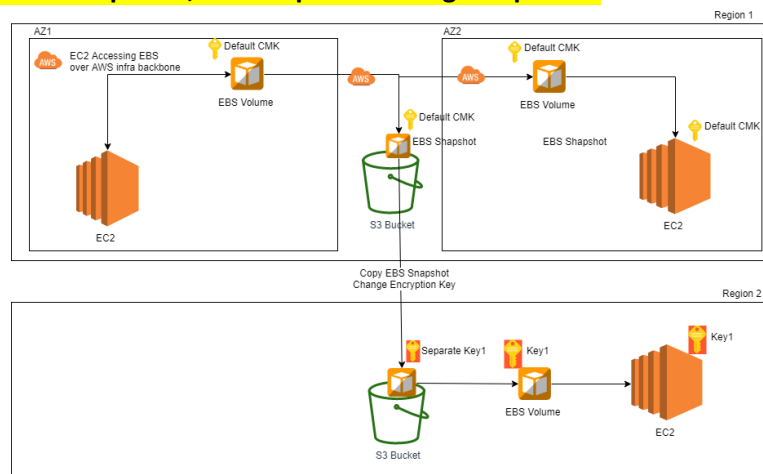
- When encrypting the first EBS volume, AWS generates a default CMK (Default Customer Managed Key). Any encrypted snapshot created from the first encrypted volume or the any image created from that snapshot will be encrypted using the same default CMK. Default CMK will be managed by AWS Key Management Service (KMS).
- Any newly created encrypted volume after first encrypted volume will have their own unique/separate AES256 bit encryption key. Which will be use in encryption of volume, its snapshot or any other volume created from the snapshot.

- In order to change the encryption key, one need to create a copy of the snapshot in the process encryption key can be changed.

- To use an EBS volume outside AZ, one need to create an EBS snapshot. EBS snapshot are region specific all AZ in the region will have access to the EBS snapshot to create EBS volume from the snapshot.

- To move/migrate an EBS volume from one region to another region, create an EBS snapshot of the EBS volume and copy the snapshot to the desired region and create a new EBS volume from the copied snapshot.

EBS volume are AZ specific, EBS snapshot are region specific.



- One cannot share a snapshot with other AWS account encrypted with default CMK, as the default CMK are assigned to a customer and cannot be exported from the AWS account.

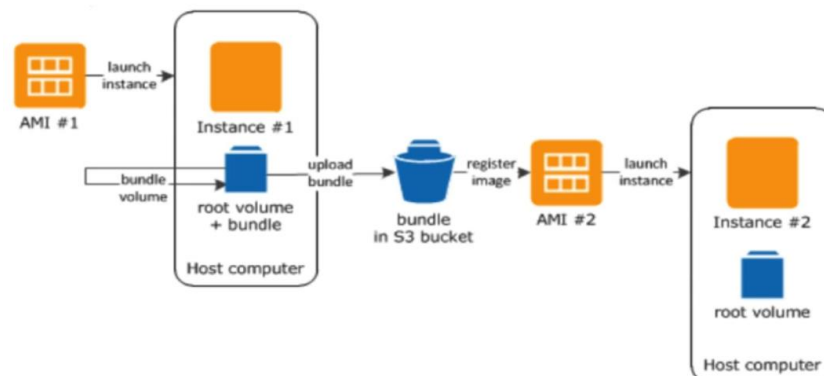
[One can share encrypted snapshot with the other AWS account, when encrypted with different key other than default CMS key.]

- **Sharing snapshot:**

- **Unencrypted snapshot** can be share with all AWS accounts by making them as public.
- Encrypted snapshot cannot be shared with all AWS accounts – there is no option to make encrypted snapshot public.
- To share an encrypted snapshot with other AWS account one need to
 - a) Ensure the snapshot is encrypted using non-default CMK. Default CMK cannot be share with other AWS account.
 - b) Provide **Cross-Account-Permission** in order to provide access to the CMK with which the snapshot is encrypted.
 - c) Mark the snapshot as private and provide the AWS account ID with whom the snapshot needs to be shared.
 - d) The target AWS account with whom the snapshot is shared needs to create a copy of the snapshot. ONLY from the copy snapshot they can re-create an EBS volume in their account.

[Recommended Best Practice]: When copying an encrypted snapshot from other AWS account, it's always advisable to change the encryption key so that copy snapshot and its EBS volume are NOT dependent on another account CMK (customer manage key). So that in future if **Cross-Account-Permission** is revoked, the snapshot and the EBS volumes are still remain accessible.

- Copying snapshot is used for:
 - Moving a snapshot from one region to another region.
 - Changing the encryption key of the snapshot.
 - Copy import/export service, AWS marketplace, AWS storage gateway.
- Use case for snapshot copying
 - Geographical expansion – extending service offering from one region to regions.
 - For Disaster recover setup – setting up remote location for disaster recovery.
 - Data retention and backup.
 - Changing region – moving service offering from one region to another region.
 - For encrypting an unencrypted snapshot.
 - For changing encryption key.
- Other key pointers:
 - ✓ While copying snapshot the tag associated with the original snapshot does not get copied.
 - ✓ While copying snapshot from a different AWS account, if the account does not have **cross-account-permission** then the copy process will **“fail silently”**.
 - ✓ At a time up to 5 snapshot copy can be initiated.
 - ✓ While the copy is in progress EBS volume cannot be created out of the new snapshot.
 - ✓ When a CMK is deleted a retention period of 7 days is enforced to ensure the key is not use elsewhere. Once a key is deleted anything encrypted using that key will be unusable.
- Creating AMI
 - Benefit of creating on the AMI
 - a) Standardization/Baselining of the EC2 images across organization.
 - b) Reduce manual effort of creating an EC2 image from basic AMI, OS patching, software installation, software configuration etc.
 - Steps to create Instance store backed AMI (root device as instance store backed)
 - a) Create an EC2 instance from AWS Instance store backed AMI
 - b) Update the EC2 instance with required patching, software installation and software configuration.
 - c) Create an AMI from the EC2 instance, for instance store back EC2 instance the AMI copy will be stored in customer define S3 bucket.
 - d) Instance store backed AMI needs to be manually register so that further EC2 can be created from the AMI instance. Since the instance store backed AMI are stored in the S3 bucket, S3 storage charges will be applicable, until AMI is deregistered and all objects related to that AMI are deleted from the S3 bucket.



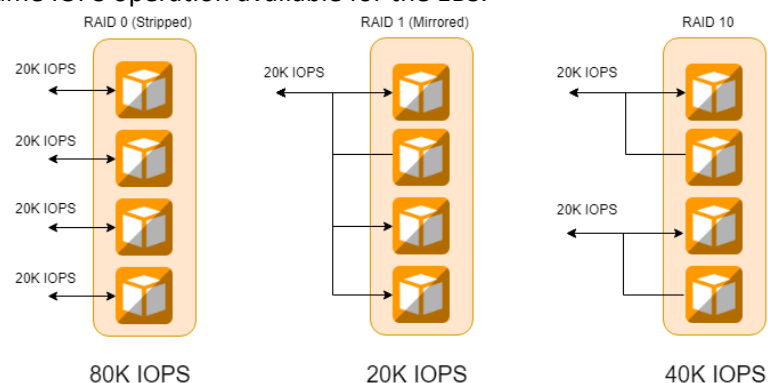
- e) Once an Instance store backed AMI is de-registered, the EC2 instances are created prior to the de-registering of the AMI instance will NOT be impacted.
- Steps to create EBS store backed AMI (root device as EBS store backed)
 - a) In-order to create an AMI from the EBS backed instances, one need to first freeze I/O for that instance by detaching the EBS volume from the instance or by stopping the

EC2 instance. EBS snapshots are point in time snapshot, to create 100% consistent snapshot one need to freeze I/O. In doing so one need to keep in mind that if there are any data stored in the instance store volume attached to the EC2 instance that data will be lost and will not be available on the AMI.

- b) When creating AMI from an EBS backed EC2 instances, AWS will automatically register the AMI unlike in case of Instance store backed AMI user needs to manually register the AMI.
- c) In case of EBS backed AMIs, user don't need to specify the S3 bucket where the AMI or its EBS snapshot are stored. AWS does this on behalf of the user.
- d) In order to delete the snapshot of the EBS backed AMI, one need to first de-register the AMI, then delete the EBS snapshots of the AMI. Deregistering of the AMI will NOT automatically delete the EBS snapshots attached to the AMI. User needs to delete the EBS snapshots manually post deregistering of the AMI.

○ Configure RAID Array

- Combining multiple EBS volume to create a single EBS volume for better I/OPS and larger disk space can be achieved through RAID Array.
- EC2 IOPS is dependent on NIC and the connected EBS's IOPS. By combining multiple EBS volume in a RAID Array higher IOPS can be achieved, which leverage multiple NIC card together to transfer data at higher IOPS. RAID is supported at the OS level, EBS volume supports any type RAID Array.
- RAID Array IOPS is dependent upon EC2 instance supported IOPS, and individual EBS volume supported IOPS.
- It's NOT advisable to use RAID array as ROOT volume of the EC2 instance.
- Different Type of RAID array are
 - Stripped (RAID 0) [**Higher IOPS, NO Redundancy**] – where multiple volumes of EBS are connected in parallel to improve IOPS. The resulting IOPS is the sum of the all connected EBS volumes IOPS. However, there are NO redundancy build on the striped RAID, single disk fails entire RAID Array fails.
 - Mirrored (RAID 1) [**No IOPS improvement, Greater Redundancy**], where multiple EBS connected in an ARRAY and data is written to all the EBS volumes in parallel.
 - RAID 10 [**Improve IOPS with redundancy**] combination of both RAID 1 and RAID 0 which takes benefit of Stripping along with mirroring.
 - RAID 5 and RAID 6 are not recommended by AWS, as the parity write operation of these models consume IOPS operation available for the EBS.



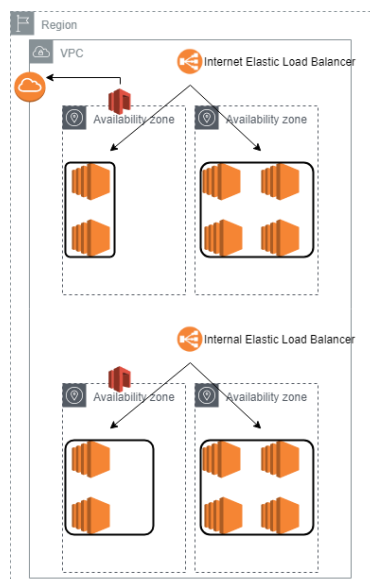
ELASTIC LOAD BALANCER (ELB)

- Elastic Load Balancer is an AWS service that helps to balance network traffic coming from internet to EC2 instance. There are three types of Elastic load balancer – *Network Load Balancer*, **Classic Load Balancer** and **Application Load Balancer** (Layer 7). From the associate exam prospective – its classic load balancer that is in scope.
- Elastic load balancer caters to the following traffic protocols - HTTP/HTTPS/SSL/TCP IP. NOT all traffic coming to the instance not necessarily needs to pass through the elastic load balancer.

- ELB Network Load balancer support – TCP/SSL – Layer 4 protocol and ELB Application Load Balancer supports HTTP/HTTPS – layer 7 protocols.
- ELB Listener connected to the backend servers – they also monitor connection request to ensure that the connection to the backend server or services are healthy.
- ELB are charged hourly basis for the service they provided, once the status change to in-service the ELB charging starts. In order to avoid any charges, incur from ELB one need to delete ELB however deleting ELB does not deletes the backend EC2 instances or the EBS volumes.
- ELB route the traffic to the primary Ethernet address (eth0).
- ELB health check – ELB monitor the health of the register instances to which it configured to route its request. ELB continuously monitor the register instance health till the time they are healthy its routes the request to its backend request, once health checks fail, ELB stops forwarding request to that instances. Healthy instances are represented as “in-service” instances, un-healthy instances are represented as “out-of-service” instances.

ELB configuration	Description	Default value	Configure Range
Response Timeout	Time by when the ELB should receive HTTP 200 status for the health check	5 Sec	Can be configure any time between 2 sec to 60 sec.
Health Check Intervals	Intervals between two probs	30 sec	Can be configure any values between 5sec to 300 sec.
Unhealthy Threshold	Number of consecutive failed probing after which an instance will be marked as “out-of-service”	2	Can be configure any value between 2-10
Healthy Threshold	Number of consecutive failed probing after which an instance will be marked as “in-service”	2	Can be configure any value between 2-10

- **Cross Zone Load Balancing:** By default, this feature is **disable**, thus ELB by default send equal load to all the zone register to it. Once Cross Zone Load Balancing is enabled ELB will consider the number of register healthy EC2 instance available in each availability zone before routing its request to EC2 instances.
- **Elastic Load balancer is a region-specific service.** Its ONLY operates in a single region.

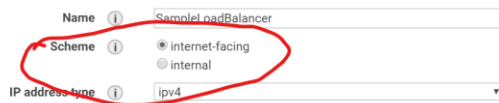


- **There are two types of ELBs – Internet facing and internal ELBs**

Step 1: Configure Load Balancer

Basic Configuration

To configure your load balancer, provide a name, select a scheme, specify one or more listeners, and select a network. The default configuration is an Internet-facing load balancer in the selected network with a listener that receives HTTP traffic on port 80.



- **Internet facing.**

Internet facing ELB are the one that are connected to the IGW and have public IP addresses. This can be reach from internet. Internet facing EBL forwards the incoming request from internet to EC2 **private IP addresses**. Its need one public subnet in each of the availability zone to route ELB internet traffic to the EC2 instance.


- **Internal facing ELB.**

Internal ELBs will have private IP addresses, and its route incoming VPC traffic to the IPv4 private address of the EC2 instance. If the instance has more than one IP address (or NIC) then ELP will route the internet traffic ONLY to eth0 NIC configured IP addresses.

Step 1: Configure Load Balancer

Availability Zones

Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availability Zones only. You can specify only one subnet per Availability Zone. You must specify subnets from at least two Availability Zones to increase the availability of your load balancer.



You are creating an internet-facing Load Balancer, but there is no Internet Gateway attached to these subnets you have selected: subnet-096929cc382ed8760

At least two subnets must be specified

[Cancel](#) [Next: Configure Security Settings](#)

ELB will always need to have a **security group** attached to it which allow communication from/to the client and the backend EC2 instances. Since ELB are define in a subnet it's important that inbound and outbound connected to/from ELB are not blocked by NACL.

ELB-Security Group Rules (internet facing)

Type	Protocol	Range	Source	Description
Custom TCP	TCP	80	0.0.0.0/0 (open to internet)	Allow all inbound transaction to EBL from internet on 80 port where ELB is configure to listen.
Custom TCP	TCP	80 (health check port)	EC2-Security Group	The reply sends back from the EC2 instance health check port needs to allowed to enter.

ELB-Security Group Rules (internal)

Type	Protocol	Range	Source	Description
Custom TCP	TCP	80	VPC CIDR block address	Allow all inbound transaction to EBL from any VPC address on 80 port where ELB is configure to listen.
Custom TCP	TCP	80 (health check port)	EC2-Security Group	The reply sends back from the EC2 instance health check port needs to allowed to enter.

Since ELB forward the inbound request to the EC2, it's not possible for the EC2 instance to know who the actual sender is. In order to know the original sender ELB support two features.

Layer 4 (SSL & TCP protocol)	Layer 6 (HTTP and HTTPS protocol)
By enabling proxy protocol feature of the ELB, one can configure the ELB to carry forward the original connection details request like original user source IP, source port, etc...	HTTP header X-FORWARD will be used to store the original connection details request like original user source IP, source port, etc. In order to know the original user details backend instance should be configure to read X-FORWARD header details.

EC2-Security Group Rules (Internet/ internal)				
Type	Protocol	Range	Source	Description
Custom TCP	TCP	80	ELB-Security Group	Allow inbound transaction from ELB security group towards port 80.
Outbound Rule				
Type	Protocol	Range	Destination	Description
Custom TCP	TCP	Ephemeral-port range (1-65535)	ELB-Security Group	EC2 reply going back to the EBL. Since ELB node can be listening to any available ports - Ephemeral-port range needs to be mentioned

- **ELB Encryption** : ELB can forward the encrypted request as is to the backend instance to achieve end-to-end encryption, also it can be configure to offload the SSL encryption from the backend instances, in that case the SSL connection will be terminated at the ELB and message is decipher and forward to the backend server over SSL or non-SSL connection as configured.
- **ELB Access Logs**: by default, the access logs are disable for ELB. Once they are enabled, they can be configured to send access log to user define S3 bucket which will be in the same region as that of the ELB. Storage cost will be applicable.
- **Sticky Session (also called as Server Affinity)**: By turning on the server affinity, ELB binds all the client session/request to a specific EC2 instances ONLY. IT required SSL termination on the ELB. *The duration of the Sticky session is determined by the expiration defined in the cookie.*
 - If application can maintain its own cookies, then EBL can be configured to maintain the expiration defined in the application generated cookies.
 - If application don't support cookies, ELB can be configure to maintain its own cookies. EBL will generate its own cookies and pass the same to the user along with its cookies. Till cookies don't expires ELB will continue to pass the request to same backend server which has server the request before, once the cookies expire ELB can forward the request to any of its configured EC2 instances.

Pro & Corns of using Sticky session:

Pro: Client session will be maintained; they don't need to authenticate again.

Corn: As the client session will be maintained in a single backend EC2 instances, if the backend EC2 instances fails the session information is lost when ELB forwards the request to another backed EC2 instance.

SSL Session Negotiation within ELB:

- Http on SSL = Https
- For front end SSL negotiation there are two possibilities
 - Define custom policy (SSL Protocols | SSL Ciphers | Server Order Preference)
 - Use pre-define policies
- For backend SSL negotiation there is on option available
 - Use pre-define policies
- SSL protocol supported are TSL 1.0, TSL 1.1 TSL 1.2 and SSL 3.0 (TSL1.3 and SSL 2.0 are not supported yet)
Note: ACM (Aws Certificate Manager) uses RSA cipher for encryption, if one is planning to use ACM for X.509 certificates then RSA Cipher should be included)
- Server Order Preference will define weather server or the client will have more preference – if enable then, then the first match on the ELB cipher list with the client list will be use. For pre-define polices this property is enabled.
- Single X.509 certificate can be loaded into ELB. In there are multiple sites then create separate EBL for each of those websites and upload their corresponding certificates.
- **ELB does not support client-side authentication (two-way authentication) with HTTPS, workaround for this is to configure ELB to work with TCP protocol instead of HTTPS and enable proxy protocol and let the backend instance (EC2 instance) read the proxy headers and perform all the task related to client-side authentication. [For this one need to ensure that EC2 instance should be able to read proxy headers and have access to server certificates & sticky session is disable on the ELB].**
- **Connection Draining** (disable by default) – AWS will wait for the in-flight instances to complete, before AWS marking the backed instance as un-healthy. When an instance is in process of deregistering with connection drain feature turn on the instance appear as *"In Service: Instance deregistration in progress."*. NO new connection will be created by the ELB during that time. Default value is 300 sec max time 3600 sec.

- **ELB monitoring:**
 - **AWS CloudWatch:** (Enable by default) ELB service will send ELB metric every one min to CloudWatch when ELB is having request passing through it during ideal time it wouldn't send any metrics to AWS CloudWatch. AWS CloudWatch metric can be use to trigger SNS notification in case threshold is reach.
 - **Access Logs:** (Disable by default): By enabling the access logs one can capture connection details like requester IP/Port, request time, request type etc., which can be stored in S3 bucket. There is no additional cost involved for enabling access logs – for storing access log information in S3 one need to pay for the storage.
 - **CloudTrail** (Disable by default): All API call made to the ELB are sent to CloudTrail. By enabling CloudTrail, all API call made to ELB can be capture. CloudTrail log can be routed to S3 bucket, by doing so S3 storage charges will be incur.
- **ELB connection time-out (ideal connection time out):** When setting an ELB connection time out, one need to consider application session timeout. If the application session time out is less than the ELB connection time-out then the backend end instance will force disconnect the connection, this may result marking the instance as un-healthy as ELB connections are getting timeout. Best practice to match the application timeout with the ELB timeout.
- **ELB Scaling:** AWS manage ELB, it takes care of the scaling of the ELB node. ELB needs 2-7 min depending upon traffic to add ELB nodes. During the time its scaling, it will NOT queue the incoming request instead return HTTP 503 error. ELB can scale if a) traffic is increasing in 50% steps in every 5 min OR b) load is increasing linearly.

Alternative scaling option

- **ELB pre warming:** When anticipating a large node, one can contact ELB make them aware of the anticipated load, so that AWS can add appropriate ELB nodes in advance to cater the increase in the traffic. This is known as ELB pre warning.
- Route53 maintains the list of ELB node that are added to the ELB DNS name, every time a new node is added/remove route53 update its list which is an authoritarian DNS for the ELB endpoint. Client will query the DNS and cache it locally for future interaction. While doing load testing one need to keep it mind of DNS Resolution else irrespective of having new node added to the ELB client will send its request to the cached ELB node address.

Best practice while doing load testing for ELB frontend services

- Use multiple test clients for load testing ELB
- Use global test sites
- In case, one test client is available for testing – then testing tool should be capable of DNS Resolution to ensure all request does not ends up to a single cached ELB node.
- When to use ELB and when to use Route 53

ELB	Route 53
Use for load balancing between instance within multiple availability zone.	Use for load balancing between regions.
ELB does not maintain cache, any instance becomes un-healthy it stops sending traffic to that instance.	Route 53 DNS service cache the request – once the instance becomes unhealthy it still forwards the request till cache data expires.

AUTO SCALING (AS)

- AWS service that enable compute to grow or shrink based on the needs is called as Auto-scaling. This will have to build a fault torrent highly available system. **Auto scaling is a regional service**, it can't work across region. They also need to be in the same VPC.
- When need to add more compute - autoscaling: SCALE OUT.
- When need to reduce compute - autoscaling: SCALE IN.
- Different components of autoscaling
 - **Launch configuration** – EC2 template which will be use when scale in is required. It includes – instance family, instance type, volumes, volume type, AMI, Keypair Block devices, security groups.
 - Launch configuration once created cannot be edited, every time it needs to be created from the scratch.

- Launch configuration can be created from AWS CLI, AWS Console or from a running EC2 instances.
- When creating a launch configuration from EC2 instances following things need to be kept in mind:
 - a) The AMI used for creating the EC2 instance should be available. (it should not be deleted or unshared.)
 - b) EC2 instance tags & any block store volume added to the EC2 instance after its launch will not be part of the EC2 instance.
 - c) EC2 instance should belong to the same auto scaling group.
- **Autoscaling group** – logical grouping of the autoscaling EC2 instances. This can be changed even after its creation.
 - Minimum Size – minimum number of instances to start with.
 - Desired Capacity – The ideal size
 - Maximum Size – maximum number of instances beyond which it can't grow.

In an auto scaling group, one can decide which subnet to be used for scaling in the instances.

If the requested instance family is not available on the targeted subnet then auto-scaling will try to create the same instance in another subnet which is defined in the auto scaling group.

- **Scaling Policy (Plan)**
 - ON-Demand/Dynamic scaling
 - Cyclic/Scheduled Scaling

When load is known/predictable cyclic(scheduled) scaling policy is used.

When scaling is needed to react to an undesired scenario then use dynamic (on-demand) policy.

- **Autoscaling rebalancing**: Autoscaling will try to evenly distribute the instances across all its Availability zones. In order to do this, it will first create instances in the AZ where it has less instances, once those instances are alive it terminates instances from the AZ where it has more instances. (*During autoscaling rebalancing, autoscaling can go 10% higher than the maximum size OR 1 EC2 instance whichever is higher*).

Why balancing is required:

- If an autoscaling group could not launch the desired instance in AZ due to unavailability of the instance capacity.
- Autoscaling was altered at a later point of time and new AZ were added.
- Manually terminated instance of one AZ.
- Manually adding instance to one AZ.
- Spot instances were terminated due to an increase in the market price.
- **What is the difference between standby instance AND terminated/de-attaching instances?**
 When an instance is in standby state – instance will still be charged as “in-service” state however they will not be part of autoscaling health-check or will have any active workload. The best practice to troubleshoot an EC2 instance which is a part of an autoscaling group is to mark the instance as “standby” instance, troubleshoot the instance and then move it back as “in-service” instance.
 Note: During a very limited timeframe after auto scaling marked an instance for replacement and before instance is terminated by auto scaling group, one can use AWS CLI (command line interface) to set the instance health as healthy (as-set-instance-healthy) to avoid termination.
- In case of **replacement** – instance is terminated first and then replaced by a new healthy instance.
- In case of **re-balancing** – new instance is added to the desired AZ, once its healthy instance from overpopulated AZ will be terminated.
- One or more ELB can be attached to an attached auto scaling group. Once attached, EC2 instances will be automatically attached to the Auto Scaling group defined ELBs. Based on the policy defined, instances will be added or removed from the ELB. ELB will be the focal point of the attached auto scaling groups EC2 instance. If connection drained is defined for the ELB, the auto scaling group will honor it.

- Auto scaling can be defined to listen to both – EC2 Health check and ELB health checks.
- If **Health Check Grace Period** is defined in the auto scaling, auto scaling group will wait till the grace period is over.
- Auto scaling group for spot instances: *[one can't mix and match spot instance and on demand instances]*.
- For spot instance, in order to change the bid price of a spot instance, new to create a new Launch Configuration OLD launch configuration can't be use.
- If auto scaling group fails to launch an EC2 spot instance in an AZ due to the market price more than that of the bid price, then it will create the spot instance in an AZ where market price is less than that of define bid price in the launch configuration. Once the targeted AZ price drops below bid price, auto scaling group will first create an instance on the targeted AZ and then rebalance the instances with the other associated AZ.
- SNS notification can be send to Emails auto scaling group IF
 - Instance is launched successfully
 - Instance is terminated successfully
 - Instance failed to launch
 - Instance failed to terminate

Autoscaling group uses CloudWatch to send SNS notification under above mention events.

- A new instance can be added to an existing Autoscaling group (AS) provided
 - Instance is in RUNNING state (i.e. its not in STOPPED or TERMINATED state)
 - The AMI for the instance is available on the region where AS exists
 - Instance should NOT be a part of another Auto Scaling group.
 - Instance should be in the same AZ as that of the Auto Scaling group.

NOTE: If the autoscaling is already operating in the maximum capacity, then the addition of the new request will be failed as it cannot be exceeded.

- Merging Autoscaling Groups: Auto scaling group can be merged using CLI, in order to merge an auto scaling group to another – define the new auto scaling group in the other auto scaling group AZ (rezone the auto scaling group to span another AZ where the second auto scaling group is connected). Then delete the second auto scaling group.

To merge separate single-zone Auto Scaling groups into a single group spanning multiple Availability Zones, rezone one of the single-zone groups into a multi-zone group. Then, delete the other groups. This works for groups with or without a load balancer, as long as the new multi-zone group is in one of the same Availability Zones as the original single-zone groups.

- Autoscaling policy
 - Scale-Out
 - Scale-In
- There are three things to auto scaling policies

Minimum capacity	Desired capacity	Maximum capacity
Minimum capacity.	Idea capacity	Maximum allowed instance at any given point in time.

- Scaling Process – there are two main scaling processes Launch and Terminate
 - **Launch** – Adding new EC2 instances to the Autoscaling Group
 - **Terminate** – removing / terminating EC2 instances from the Autoscaling group.
 - **AddToLoadBalancer** – Adding new instance to ELB or Target Groups.
 - **AlarmNotification** – Accepts alarm notification from the cloudWatch that are associated with the Autoscaling policies
 - **AZRebalance** – Balance EC2 instance within the Autoscaling connected AZ – evenly balancing the EC2 instances in all the associated AZs.

- **HealthCheck** – Checks and validated the health of EC2 instances based on the information received from EC2 service or ELB services.
- **ReplaceUnhealthy** – Terminate unhealthy instances and create new instance based on the Launch template.
- **ScheduledActions** – performs proactive scaling – based on the provided schedule.

NOTE : Any of the above mentioned process can be suspended and resume later, suspending process may also lead to preventing other process to work properly.

○ Scaling Policy

- **Manual Scaling** - Manually adding/removing EC2 instances to maintain a desired count of EC2 instances. This can be done by manually changing auto scaling group min/desired/max values, or by attaching/detaching instances manually.
- **Cyclic (Schedule base) scaling [for predictable load change]** – When the load is predictable one can schedule to add/remove more instances to cater the growing/shrinking load.
 - a) Each of the schedule scaling policy should have unique date-time (unique start & end date). For example: Before black Friday sale.
- **On-Demand (Event base scaling) [for unpredictable load change]**– when EC2 instances needs to be added base on an event (dynamic load) then use On-Demand policies.
 - a) Alarm can be created on a single metrics. Cannot be created based on multiple metrics.
 - b) Cool-down-timer (default value 300): A time gap after acting on a scaling activity to give sufficient time for the EC2 instances to be provision and come to in-service state.
 - c) There are two types of on-demand scaling
 - *Simple Scaling* - adding capacity in single shot
 - *Step Scaling* – adding capacity in multiple steps. In case of step scaling cool down period will not be applicable, instead **warm up time** is applicable: After the scaling action is trigger time gap before proccing a new alert.
- **Single auto scaling group can have multiple scaling policies attached to it at a given time.**
- **Auto scaling cannot change the capacity of the group above the maximum capacity OR below the minimum capacity.**
- Default Scaling policy is **target scaling policy**: Base on one of the below metrics auto scaling will occur. Once set target is reached auto scaling policy will kick in.
 - a) Application Load Balancer Request count per target
 - b) Average CPU utilization
 - c) Average Network in (bytes)
 - d) Average Network out (bytes)

One can disable scale-in feature to prevent instances from auto termination. Also, one can set warm up time, till warm up time counter is not ZERO another policy will not be triggered.

- Auto scaling monitoring: As a part of the launch configuration, one can define EC2 instance monitoring. If the Launch configuration is created from
 - a) AWS console then by default basic monitoring is activated (which collects & send metrics to CloudWatch every 5 min).
 - b) AWS CLI then by default detailed monitoring is activated (which collects & send metrics to CloudWatch every 1 min).
- Apart from collecting & monitoring EC2 metrics, auto scaling group can also send aggregated metrics of the auto scaling group (*which also includes EC2 instances metrics*) to CloudWatch. By default, auto scaling group aggregated metrics will be sending every one min.
- To changing monitoring option in auto scaling group, a new launch configuration with the desired monitoring option needs to be recreated (as Launch Configuration can be edited, can only be copied or create fresh).

- The launch configuration monitoring setting AND the auto scaling group policy setting should match – if its basic monitoring then both needs to be basic monitoring/ if its detailed monitoring then both needs to be detailed monitoring.
 - a) If EC2 instance monitoring set to basic in Launch configuration – set auto scaling alarm to 300 sec.
 - b) If EC2 instance monitoring set to detailed in Launch configuration – set auto scaling alarm to 60 sec.

If there is a mismatch in the auto scaling monitoring & EC2 monitoring, as such there will not be any error, however if EC2 instance is set as detailed monitoring and auto scaling is set as basic monitoring then 4 out of 5 EC2 metrics reading will not be used.

RELATIONAL DATABASE SERVICE

- There are two primary purpose of a databases
 - OLTP (online transaction processing).
 - OLAP (online analytical processing).
- There are two types of databases
 - **Relational Database (RDMS)** [structure database]
 - **Non-Relational Database (Non-RDMS)** [un-structure database].
- Different types of non-relational databases
 - **Columnar Databases:** These are optimized for reading columnar data, Example: Apache HBase, Apache Cassandra]
 - **Document Databases** [These are used for storing non-structural document base objects in SON or XML data. These types of databases are fit for storing varying structure data and where high read/write performance is desired at lower cost. Common use cases are: gaming application / web application / mobile application / IoT applications / Ad Tech etc.]
 - **In-Memory databases key-value store:** These are optimized to store key-value which demands heavy workload (frequent reads). These types of databases are fit for heavy workload application like – recommendation engine.
 - **Graph Databases:** A graph database is a database that uses graph structures for semantic queries with nodes, edges, and properties to represent and store data. A key concept of the system is the graph (or edge or relationship). The graph relates the data items in the store to a collection of nodes and edges, the edges representing the relationships between the nodes. The relationships allow data in the store to be linked together directly and, in many cases, retrieved with one operation. Graph databases hold the relationships between data as a priority. Querying relationships within a graph database is fast because they are perpetually stored within the database itself. Relationships can be intuitively visualized using graph databases, making them useful for heavily inter-connected data. (Neo4J and Amazon Neptune are example of graph databases.)
- AWS manage following service for RDS instances
 - Hosting of DB instances
 - Security & patching of instances
 - Automated backup of the DB instances
 - Software updates of DB instances
 - **Synchronous replication of DB instances across multiple AZ** (*this needs to be explicitly configured on the DB instances by enabling multi-AZ feature*)
 - Provide ability to create read replicas to optimize performance of heavy read operation databases.
- AWS doesn't manage following service for RDS instances
 - Managing DB settings
 - Building DB schema for non-relational databases
 - DB Performance tuning
- List of AWS managed RDS databases

1. MS SQL
 2. Oracle DB
 3. PostgreSQL
 4. MariaDB
 5. AWS Aurora
 6. MySQL
- Licensing model supported in AWS
 - BYOL (bring your own licensing model)
 - Licenses included
 - **Note: Max of 40 db instances can be created per AWS account.**
 - **10 out of 40 can be of Oracle or MS SQL database type, when opted for Licenses included model**
 - **All 40 DB instances can be of type Oracle or MS SQL database type if BYOL model is opted instead of Licenses included model.**
 - **Amazon RDS uses EBS volumes as databases block store**
 - **General Purpose RDS storage** is mainly suited for DB workloads for moderate I/O requirements.
 - **Provision IOPS RDS storage** is mainly use for high performance OLTP workload.
 - **Magnetic Disk RDS storage** is ONLY use for smaller databases.
 - Maximum capacity of the RDS storage capacity
 - ~~Up to 4 TB for MS SQL server~~
 - ~~Up to 6 TB for other RDS engines.~~
 - ~~MS SQL storage limit is because of the limitation of striped storage that can be attached to the Microsoft windows server.~~
 - Maximum size of the AWS RDS DB instance is 16 TB
 - Multi AZ configuration
 - This can be enabled when db instance is getting created.
 - When enabled, an instance of the DB will be created on a separate AZ within the same region.
 - The secondary (standby) instances will be automatically synchronized with the primary (main) instance.
 - NO read/write can be done on the secondary (standby) instances while primary db instance is still active.
 - **AWS does not provide option to select AZ when enabling multi-az option in the database. However, once created one can view the standby db instance AZ.**
 - Depending upon the instance type, it can take 1 to few minutes for the DB instance to failover.
 - AWS recommended **provision-IOPS for multi-AZ base set up.**
 - In case of Multi-AZ setup, the failover will be triggered on the following scenario:
 - Automated failover trigger**
 - Loss of Primary database AZ or Primary DB instance failure.
 - Loss of Primary instance network failure
 - Loss of Primary instance compute failure
 - Loss of Primary instances storage failure
 - The primary DB instance change
 - Patching of the Primary db instance OS.
 - Manual failover trigger**
 - This can only happen when; primary instance is rebooted with *“reboot with failover”* option.
 - The following are the reason when a primary db instance needs to be rebooted:
 - When there is a change in the parameter group.
 - When there is a change in the static DB parameter.

Note: parameter group is the container for the DB instance configuration, which needs to be change to tune performance.
 - When instance is failover to work seamlessly, its recommended to refer the DB instance with its CNAME (endpoint) instead of DB IP address.

- During failure, AWS automatically update the CNAME (endpoint) with the secondary (stand by) db instance IP address.
- On an event of any RDS db instance failure, AWS will send an SNS notification (*this needs to be enabled, not enabled by default*).
 - Using *DescribeEvents* API call one can view RDS failure events of PAST 14 days.
 - Using AWS CLI, one can view RDS failure events of PAST 14 days.
 - Using AWS console one can view ONLY RDS failure events of PAST 24 Hours (1 day).
- DB instance OS level changes are done on the STANDBY first then promote the standby to PRIMARY before applying the changes to the STANDBY instance. Following are the OS level changes.
 - OS Patching
 - System Upgrade
 - DB Scaling
- **AWS can restore DB backup up to RPO of 5 min.**
(Recover Point Objective – time in past till when the instance can be restored).
(Recover Time Objective – time taken to recover).
- While taking snapshot and backup of an instance there will be a freeze in the I/O operation in case of a stand-alone instance, in case of a multi-AZ setup the snapshot and backup of an db instance will be initiated on the standby instance, to avoid any a freeze in the I/O operation in the primary instances.
- RDS versions can be upgraded to any other supported RDS version
 - User can trigger the upgrade (*modify db instance version*), and the change will take place in the next maintenance window.
 - User can force upgrade the db instance version, and the change will take place immediately.
 Note: During db instance upgrade the db instance will not be accessible for read/write operation for stand-alone as well as multi-AZ setup as both the instances will be simultaneously upgraded.
- **In case of multi-AZ setup use needs to make sure the Application is allowed to communicate to both the primary and secondary (standby) db instance subnets.**
- **RDS backup:**
 - **Automatic backup** (*enable by default, user can disable if not required*)
 - **Manual backup** (need to trigger manually)

Note: When AWS RDS backup is trigger – it doesn't just backup a database, entire db instance is backed up into a snapshot which can later be restored. And store the snapshot into multiple AZ for durability. The db instance snapshots are stored in S3.

Automated Backup	Manual Backup
Can be used for point in time recovery.	Cannot be use for point in time recovery.
Trigger automatically	Trigger Manually
When RDS instance is deleted automated backups are automatically deleted.	When RDS instance is deleted, manual backup still remains in S3 bucket. One needs to manually delete the backup.
Cannot be share directly with other AWS account.	Can be share directly with other AWS account.

There are no charges for enabling automated backup for a db instance, however the storage will be chargeable.

DB instance should be in “*active*” state for automated backup to take place, if the instance is in “*Storage_Full*” state the automated backup will not occur.

- **Using automated backup, it's possible to restore RDS DB instance point-in-time, however same cannot be achieve in manual backup.**
- **RDS automatic backup is taken daily, including DB transaction logs. Using automated backup & DB transaction logs it's possible to restore an DB instance up to 5 min in past point-in-time. One can define backup window.**

- **Note: Once DB is restored, ONLY default DB parameters and Security groups will be automatically restored – custom DB parameters and security group settings needs to be applied.**
- **Retention period** is the timeframe till when the db instance backup is retained.
 - Max retention period = 35 days,
 - Default retention period from AWS console = 7 days (**for aurora db its 1 day**)
 - Default retention period from AWS CLI = 1 day
 - When set retention period =0 automated backup will be disable.
- **For MySQL automated backup is supported only for InnoDB storage engine, currently it's not supported for MyISAM storage engine.**
- Automated backup cannot be shared within other AWS account. One need to copy the automated backup copy and share it with other AWS account.
- DB instance restore
 - When the DB instance is restored it restore back to default DB parameters, one needs to manually apply the DB parameter changes and also the security group.
 - The restore DB instance will have different endpoint then that of the original DB.
 - When restoring DB instance one can change the storage type.
- DB Subnet Group – collection on subnets where db instance can be launch. When a db instance is launch user have option to select the subnet group and a specific subnet from the subnet group. This give better control over the NACL and security group configuration. However, one cannot control in which IP address of the specified subnet the DB instance will be launch.
- RDS database encryption
 - When RDS DB instance is created one can enable encryption.
 - For already existing DB instances
 - Create a new encrypted DB instances then migrate existing (unencrypted) DB instances data to it.
 - Restore the existing (unencrypted) DB instances to a new DB instances with encryption enable.
 - When encryption is enabled, all the following will be encrypted
 - All its snapshot
 - Backups
 - Data storage (on DB instance)
 - Read replicas created from the DB instances
- RDS DB instance costing
 - DB instance hours (for partial hours it needs to pay for full hours)
 - Storage (*EBS storage*) per GB/month
 - I/O request/month (this is only for magnetic RDS instance type)
 - Provision I/O request/month (this is only for provision IOPS SSD instance)
 - Internet database transfer
 - Backup storage - automated backup storage equal to the sized of the DB instance size is FREE for the same AZ where the RDS instance is launched. AWS store the backup in multiple AZ for durability, backup stored in AZ other than the AZ where RDS is launched will be chargeable.
 - **For Multi-AZ setup additional cost will be applicable**
 - RDS instance hours for the standby database
 - Storage for the standby database
 - Additional IOPS per month for synchronization
 - Data transfer between primary and secondary is NOT CHARGEABLE.
- Reserved Instance for RDS instance
 - Reserved instance should have exact match
 - Instance Type
 - Storage
 - DB Engine
 - Standalone or Multi-AZ

- Reserved instance is region specific, cannot be move from one region to another region.
However, they can be move from one AZ to another AZ within a same region.
- **Standby instance should be made in the same region (across multiple AZ), Read replication can be done across region.**
- **The Standby and Primary has same endpoint with different IP addresses, in case of failure the IPs will be swap in DNS.**
- **On an event of an RDS instance failure, RDS will send SNS notification to RDS event category group by RDS service – one needs to subscribe to these event categories.**
- One can also set CloudWatch alarms based on certain metrics (single alarm/single metrics) which can be sent to SNS or take specific actions.
- **Read Replicas:** duplicate RDS instance of the primary db instance, which can be primarily use for scaling heavy read operation. The following DB engine supports read replicas.
 - Read Replicas use cases
 - Catering to read heavy operation
 - For supporting higher I/O demand which cannot be meet by the DB engine specification
 - Supporting read operation in case of a network failure with the primary database engine.
 - **Postgres SQL, MariaDB, MySQL, Aurora DB**
 - MariaDB, MySQL and PostgreSQL one can have read replicas in other region, ~~Aurora does not support multiple region read replicas.~~ Aurora DB supports up to 5 cross region replications.
 - One needs to enable automatic backup (*retention period not be equal to zero*)
 - For Postgres SQL, MariaDB, MySQL the max up to 5 replicas can be created for each primary database instances
 - Aurora DB supports up to 15 read replicas
 - MariaDB, MySQL supports writing in the read replicas
 - **MariaDB, MySQL supports creating of read replicas of read replicas – max up to 4 replication level can be created.**
 - In case of Multi AZ failover – one the primary DB instance failover to standby instance – read replicas automatically points to new primary DB instances.
 - While creating read replicas
 - Read replicas can be created from AWS console / AWS CLI
 - One can select the AZ where read replicas needs to be hosted
 - Read replicas storage & instance type can be different than that of the primary database storage and instance (but it should have minimum storage and compute of the primary database, if primary db instance is scale-out the read replicas should also be scale-out to meet the minimum requirement)
 - Maria DB and MySQL supports read replicas of read replicas, however maximum of 4 level can be created as chain.
 - Chain read replicas will have a greater lag.
 - While deleting read replicas
 - When DB instance is deleted – one needs to manually delete the read replicas.
 - If the primary DB instances is deleted – the read replicas get promoted as standalone primary databases.
 - While reapplication terminated
 - If more than 30 consecutive days replication is turn off (manually or due to error), AWS delete the read replicas automatically.
 - When Primary DB instance is lost with read replicas
 - Read replicas can be promoted as primary database – to reduce RPO and RTO.
 - Read replica can be promoted to standalone DB instances within a single AZ.
 - Once a read replica is promoted to primary db, the following parameters it inherited from the primary db
 - **Backup Retention Period**

- Back window

- DB Parameter Group

- Once a read replica is promoted to primary db the other read replicas of the existing primary db continues to work as is without any changes.
- InnoDB DB transaction storage engine supported for RDS replication. Non-transactional storage engine like MYISM may prevent read replicas to work as intended.
- RDS scaling
 - While scaling the RDS instance there will not be any downtime except
 - When upgrading the db engine (applicable even to multi AZ setup)
 - When changing db parameters (applicable only to standalone db instances NOT applicable to multi AZ setup).
 - RDS storage can only be increase
 - RDS storage type can be change (except MySQL)
 - RDS scaling can be set as apply immediately OR apply during change window.
- RDS Pricing
 - If an RDS engine is launched, regardless whether it's used or not it will be chargeable to the customer.
 - Licensing option in RDS are BYOL and License included.
- **RDS Enhance monitoring:** For the following RDS engine, one can turn on the RDS enhance monitoring feature which will provide real time metrics from the OS Level through the agents deployed on the instances where the DB instance is running. These may differ slightly then those RDS CloudWatch CPU utilization metrics which are collected at the hypervisor level as small hypervisor overhead will be added to the Metric collected. For Multi-AZ configure RDS instance, enhance monitoring show result of both Primary as well as of the failover instance.
 - MariaDB
 - Microsoft SQL Server
 - MySQL version 5.5 or later
 - Oracle
 - PostgreSQL
- RDS Enhance monitoring process list are categorized under
 - RDS Process: Summary of the RDS Process, that are used by the RDS Management agent, diagnostic monitoring process, and other AWS processes that are required to support the RDS instance.
 - RDS Child Process: Summary of the child process that support the RDS instance.
 - OS Process: Summary of the Kernel and system level processes.
- **RDS Performance insight:** extend the RDS monitoring illustrate the DB performance and analyzing issues by visualizing the load and filtering the load based on the wait, SQL statement, host and user.

- Amazon Arora DB:

Amazon Aurora employs an SSD-backed virtualized storage layer purpose-built for database workloads. Amazon Aurora automatically replicates your storage six ways, across three Availability Zones. **Amazon Aurora storage is fault-tolerant, transparently handling the loss of up to two copies of data without affecting database write availability and up to three copies without affecting read availability.** Amazon Aurora storage is also self-healing. Data blocks and disks are continuously scanned for errors and replaced automatically. For more information about high availability with Amazon Aurora

- Managed DB service from AWS – which is compatible with PostgreSQL and MYSQL
- It can grow up to 64 tera bite with minimum size of 10 GB.
- Aurora DB cluster consist of multiple DB instances – group under a cluster and can be access through cluster endpoint or through instance endpoints.
- The data are stored under cluster volume which are stored under multiple AZ.

- Each Aurora DB cluster has one primary db instances – which can be used for performing read & write operation. Aurora cluster can also have up to 15 read replicas, which offload read workload from the primary db instances & if the primary instance fails it acts as failover.
- Aurora DB endpoints:
 - **Cluster Endpoints:** each Aurora DB has one single primary instance and one cluster endpoint – this endpoint can be used for both read and write operation.
 - **Reader Endpoints:** This is the read-only endpoint which load-balanced incoming request (query) and direct it to a specific Read replica. This endpoint can't be used for write operation.
 - **Custom Endpoint:** This helps in forwarding incoming request to a specific instance & define what kind of operation can be allowed on that instance.
 - **Instance Endpoints:** Individual end point of the Aurora DB instance, these are typically used for diagnostic capacity and performance issue on a specific DB instance.
- Aurora Redundancy:
 - Aurora DB stores cluster volume in multiple AZ.
 - Aurora DB detects disk failure and can repairs the segment on their own.
 - Post startup after failure, it copies the buffer pool to make the data readily available.
 - It can recover instantaneously from its failure.
- Aurora Fault Tolerant:
 - It synchronously synchronized primary db instance with another read replicas instance.
 - When primary instance fails the following things happens

In case of the single Aurora DB instance	In case of DB instance with read replicas	In case of serverless instances
<p>It will try to create a new DB instance in the <i>same availability zone</i> as that of the original DB instances and replace the original instances with the new instance.</p> <p>IF failed it will try to attempt to create a new DB instance in a different availability zone.</p>	<p>It will flip the CNAME to a healthy read replica instance.</p>	<p>It will create a new DB instances <i>in a different availability zone</i> as that of the original instances.</p>

- Data blocks are continuously scan for error and repairs automatically.
 - Aurora MYSQL supports cross region replication either by physical or logical replication.
 - Aurora PostgreSQL doesn't support cross region replication
- Aurora DB configuration
 - Based on the DB instance classes Aurora DB performance are determined, there are two types of instance class available for Aurora DB instances
 - **Memory Optimized** – Optimized for memory intensive operation.
 - **Burstable performance** – were instance are configured to burst full CPU usages.
 - **Aurora DB Serverless configuration** which is an on demand autoscaling configuration for the Aurora DB compatible with MySQL and PostgreSQL. This can startup, shutdown and scale as per the need. One need to define
 - Maximum Aurora Capacity unit: Maximum limit.
 - Minimum Aurora Capacity unit: Minimum limit.
 - Pause after inactivity: the amount of time with no DB interaction the DB instance can be scaled to ZERO.
- **Aurora DB Reboot** – NO failover occurs: When a primary instance is rebooted, its read replicas are also rebooted automatically.
- **Delete Protection flag can be enabled to avoid accidental delete.** An Aurora DB instance can't be deleted if BOTH the conditions are true.
 - If the Aurora DB cluster is NOT a read replica to another DB cluster

- There is ONLY one DB instance.
- Aurora DB monitoring
 - **Subscriber to Amazon RDS Events** to know the changes occurred in DB instance, DB cluster, DB parameter group, DB cluster snapshot.
 - **RDS enhance monitoring:** Look at the metrics at real time for the operating system.
 - **RDS Performance insight:** Monitor your Amazon DB load to analyzed and troubleshoot database performance issue.
 - **CloudWatch Metrics – alarms, metrics and logs**
- Aurora DB Security
 - **Use IAM to control access**
 - **Security Group can be used to control which devices / EC2 instances can connect to Aurora DB cluster endpoints.**
 - **Configure SSL and TSL to connect to Cluster endpoint**
 - **User RDS encryption to secure RDS instance and the snapshot at rest.**

Aurora DB for PostgreSQL	Aurora DB for MySQL
Push button compute scaling	Push button compute scaling
Storage Autoscaling	Storage Autoscaling
Low latency read replicas	Low latency read replicas
Custom Database endpoints	Custom database endpoints
Scaling <ul style="list-style-type: none"> • Instance scaling – modifying instance class for better performance • Read scaling – by adding read replicas 	Scaling <ul style="list-style-type: none"> • Instance scaling – modifying instance class for better performance • Read scaling – by adding read replicas
Aurora DB for PostgreSQL supports logical data replication, data changes in PostgreSQL Aurora DB can be replicated to the other DB using native PostgreSQL slots or other data migration tools	
	Global database is currently available for Aurora DB MySQL compatible version.

SIMPLE STORAGE SERVICE (S3)

○ Block Storage Vs Object Storage

Block Storage	Object Storage
In block storage, data will be divided into smaller equal sized blocks and then stored within a block store like EBS (Elastic Block Store) which can be referred by an index by the block store.	In object store data are stored as whole. They are not divided into any smaller unit. Object up to 5 Tb can be stored as a whole. They can be referred by – date/object meta-data/filename/version number (optional).
Example of a Block store: EBS volume	Example of an Object store: S3
What can be store in block storage: DB transaction logs, software installed folder	What can be object in block storage: File, photo, video, music file, EBS volume snapshots, DB snapshots, AWS cloudTrail logs
ONLY the EC2 instance where its mounted to.	Accessible from anywhere in the world.

○ Data consistency model for distributed storage architecture

- **Immediate (Strong) Consistency model** – when a data that written to multiple system, and read immediately if the user gets same response from all the system then, it's called as immediate consistency model.

- **Eventual Consistency model** – when a data that is written to multiple system and read immediately if the user doesn't get the same response from all the system then, it's called eventual consistency model. Its preferred for
 - High Stability
 - Higher Availability
 - High Data Durability
 - Lower cost
- In case of S3 when a new object is first written (PUT) into S3 storage it will ensure **read-after-write consistency model** (immediate consistency model).
- In case of updating or deleting an existing object (override PUT or DELETE) into S3 storage then S3 will ensure eventual consistency model.
- S3 durability is 11 9's (Chances of losing 1 in 1000000000 percentage)
- **Minimum size of an object that can be stored in an S3 bucket is 0 Bytes.**
- **Maximum size of an object that can be stored in an S3 bucket is 5TB.**
- S3 bucket is non-hierarchical, i.e. one can't have another S3 bucket within S3 bucket however once can create folders within S3 bucket from web console.
- S3 bucket is a region specific – its store uploaded object in multiple location within a single region.
- **S3 bucket ownership can't be transfer from AWS account to another.**
- By default, object uploaded to a S3 bucket is private ONLY bucket owner can access the objects.
- If the object is uploaded by and IAM user, still the object is owned by the account owner NOT the IAM user.
- S3-Sub Resources
 - **Life Cycle policies:** Policies to decide object lifecycle
 - **Website:** To hold configuration to host static websites
 - **Versioning:** Maintained different version of updated objects
 - **Access Control List:** Bucket access
 - **Bucket Polices:** Bucket object access
 - **Cross region replication:** Replication across different region.
 - **Torrent:** use S3 object to be download using bit-torrent software.
 - **CORS:** Cross Origin Resource Sharing.
 - **Logging:** Bucket access logs.
- There are two types of s3 URLs
 - **Virtual HOST Type** e.g. `https://[bucket-name].s3-[aws-region].amazonaws.com`
 - **Path Type** e.g. `https://s3-[aws-region].amazonaws.com/[bucket-name]` for US East (N. Virginia) Region alone the endpoint will be `http://s3.amazonaws.com/[bucket-name]`
Starting March 20, 2019, bucket created in the region will not be reached through `https://[bucket-name].s3.amazonaws.com`.
- Data transfer charges within EC2 instances and S3 bucket within a same region doesn't incur any charges.
- S3 cross account access for object upload
 - Under cross account access, account from which the object was uploaded owns the object not the bucket owner.
 - Bucket owner still pays for the object storage, uploaded by another account.
 - Bucket owner can deny access to object which are NOT owned by them.
 - Bucket owner can delete/archive the object which are NOT owned by them.
- Whom can be granted to access S3 buckets?
 - IAM users
 - Another AWS account.
 - Authenticated Users – who are not AWS users.
- S3 access policies
 - A S3 bucket can have a Bucket Policy or bucket ACL/object ACL
 - A S3 object can have
 - Resource Based Policies
 - User Policies

- S3 Access Level Policy
 - **Account owner** – AWS resource that created the bucket. A bucket owner can delete an object regardless of who owns the object.
 - **Resource owner** – AWS account that creates the object. Object owner owns the object. If an IAM user uploads an object, it's the AWS account that has created the IAM user owns the object not the IAM user.
- Access Policies attached to S3 resources
 - **Resource base policies** – policies that are attached to a resource (bucket and objects).
 - **User base policies** – policies that are attached to a user who can access the resources.
- Resource base policies
 - ACL
 - Each bucket & object can have ACL attached to it.
 - ACL is a list of grants that identify grantees to access specific S3 resources.
 - ACL can be use only to give basic read/write grant to attached S3 resources.
 - Bucket policies
 - Bucket polices can be created to grant access to a specific s3 resources.
 - Then the AWS owner can only provide deny access to the objects that are not created by them but they can't grant read access to those objects.
 - In many case Bucket policy replace ACL policy.
- User base polices
 - Can grant access to a bucket or the any particular object to IAM users
 - Can create user, group and role which can be attached to an IAM user to grant access to S3 resources.
 - Anonymous access can't be granted using user base policies as it needs a specific user / group / role to get it attached to.
- Evaluation of an access for S3 resources
 - User context
 - S3 checks if given user has permission from the AWS account owner for IAM users.
 - S3 checks if user has access to the requested object.
 - Bucket context
 - S3 check if the bucket owner allows the use to perform the request operation on the bucket

Note

- If bucket and the object is the same AWS account then, IAM user can be granted access either by resource policy or by user policy.
- If the bucket owner and resources owner is the same – then bucket policy will be evaluated for the access.
- If the bucket and resource owner are different then -owner must use object ACL to grant permission.
- S3 ACL (bucket ACL and user ACL) can provide finite set of control – bucket policy and user policy are the one that provides granular control.
- Default ACL gives full control to the AWS account owner over the created resource (bucket or object).
- Using **ACL**, AWS account owner can grant access to the grantee AWS account, later AWS grantee account owner can delegate that access to its IAM users. Using ACL s3 bucket owner can't directly grant access to the grantee's IAM user.
- List of access that can be granted from ACL

Permission	When granted for a bucket	When granted for an object
READ	Allows grantees to list all objects within the bucket	Allows grantees to read the object and its metadata
WRITE	Allows grantees to write, overwrite, delete any object in the bucket	NOT applicable
READ_APC	Allows grantees to read bucket ACL	Allows grantees to read object ACL

Permission	When granted for a bucket	When granted for an object
WRITE_APC	Allows grantees to write ACL for the attached bucket	Allows grantees to write ACL for the attached object ONLY.
FULL_CONTROL	Provide grantees READ, WRITE, READ_APC and WRITE_APC access on the bucket.	Provide grantees READ, WRITE, READ_APC and WRITE_APC access on the object.

- Use cases for Object ACL and bucket ACL
 - Object ACL
 - Object ACL are used to grant access to an object to another AWS account who is not the bucket owner. (However, bucket owner can still delete/archive the object).
 - **Bucket Policy can of MAX 20KB size** – therefore to grant granular access at the object level it required to use object ACL. (**Object ACL can have up to 100 access policies in it, and only one object ACL is permitted per object.**)
 - Bucket ACL
 - **AWS recommended to use bucket ACL to grant access to the S3 log delivery group.**
 - Bucket ACL can be used to grant bucket policy to another AWS account
- When to use bucket policy or user policy?
 - Bucket policy (can provide partial range of permissions)
 - When AWS account owns the bucket and wants to grant access to the bucket to its users.
 - When AWS account own the bucket and object and it want to grant access to the object to its users.
 - When AWS account wants to grant an access to another AWS account to access its buckets.
 - For granting access to S3 log delivery group.
 - User policy (can provide full range of permission)
 - User policy can be used to grant full range of s3 permission.
 - To grant access to the IAM users of same AWS account.
 - To delegate the grant the it receives from another AWS account that own the bucket/object.

Note: IAM user must have access permission from the parent account where it belongs to & also from the resource owner.

- Parent owner can share its access to any resource/bucket using user policy.
- Resource owner can share its access to other account users using bucket policy OR share access to other account using bucket policy, bucket ACL, and object ACL which account later can share with its users.

AWS account shares its access to another AWS account, that account CAN'T further share the access to the other AWS account.

- **S3 versioning:**
 - Once enable cannot be disable, however it can be suspended at any given time.
 - By default, GET will return the latest from the S3 bucket
 - Delete Marker - Once a versioned object is deleted it doesn't get deleted, only a delete marker gets attached to the object. Later one can delete the delete marker and make the object re-available.
 - Once versioning is enabled, it will charge for all the version that are hosted within the S3 bucket.
 - Second level security for deleting versioning delete can be enable using MFA (Multi Factor Delete) delete.
- **Copying and Updating Object:**
 - Multi Part upload – for object size large than 100MB its recommended to use multi part upload feature instead of single upload to improve performance (*speed up the upload process*). File with size 5MB to 5TB can be uploaded using multi part upload feature. [**file size more than 5GB, multi part upload is the ONLY option.**]
 - Transferring files within region does not incur transfer charges.

- While copying storage class and encryption status can be changed.
- There are two types of metadata – system define metadata & user define metadata, some of these metadata can be changed or added new when copying the files
- When s3 upload is successful, it returns a HTTP 200 OK response message back.
- When s3 uploads is successful for a file which has requested for Server-side encryption SSE using customer provide key file along with HTTP 200 OK response it also returns encryption algorithm & the MD5 encryption key used specified during upload.

○ **Storage Classes**

Real time storage class						Archival Storage
	Standard	S3-IA	S3-Single Zone - IA	S3-Reduce Redundancy Storage	S3-Intelligent Tiering	Glacier
Sustainability	Design to sustain data loss in two facility.	Design to sustain data loss more than 1 facility.	Design to sustain data loss in 1 facility.	Design to sustain data loss in 1 facility.		Store within a three physical availability zone within a same AWS region.
Availability	99.99%	99.90%	99%	99.99%	99.99%	NO SLA
Durability	99.99999999 9 9's	99.99999999 9 9 9's	99.99999999 9 9's	99.99%	99.99999999 9 9 9's	99.99999999 9 9 9's
Minimum Size	0 KB – 5TB	Greater than 128 KB			128KB	1 Byte – 40TB
Usages	critical data	Old less frequently use data.	Non-critical data	Non-critical data / temp for downloading archive data	Use for those objects whose access pattern are complex to identify.	Archiving data
Retention Period	Minimum of 30 days charge is applicable	Minimum of 30 days charge is applicable.	Minimum of 30 days charge is applicable.		Minimum 30 days charges are applicable	Minimum of 90 days charge is applicable.

- Once requested – glacier data will be made available in 3-4 hours, the data will be retrieved and upload to a S3-RRS storage class.
- If the data is more than 4GB, then it needs to be uploaded as multi part upload. Any data more than 100 MB can be uploaded as multi part upload.
- Data can't be upload to Glacier storage class from AWS console one can ONLY use command line interface, AWS API or AWS SDK data can be upload to Glacier storage class.
- When a lifecycle policy transitions a data from a S3 storage to Glacier it adds 32KB of data for indexing and achievable metadata. This overhead increase to 40KB (extra 8KB) if the S3 is use to load the data to glacier. *[In order to keep the overhead size small, it's advisable to bundle up the data instead of uploading it in small chunks. While bundling up the data it always required to uses right compression technique where one can easily download individual files from the uploaded bundle.]*
- Glacier doesn't allow any metadata to the uploaded data except the archive description. So, it's important to maintain a client-side repository about the archive meta data.
- Once a data is uploaded into glacier, then it cannot be changed.

- Glacier data retrieval
 - **Expedite retrieval:** fastest retrieval time 1-5 minutes/ costliest among other retrieval process.
 - **Standard retrieval:** 3-4 hours of retrieval time / up to 10 GB of data retrieval is free.
 - **Bulk retrieval:** 5-12 hours of retrieval time/ Cheapest among other retrieval process.
 - **Part retrieval:** Part of archive retrieval.
- Note: Provisioning Capacity ensure that the required retrieval capacity is available when its needed.**
- Glacier data upload is a synchronous process- S3 will send success message back only when data is store across multiple location within Glacier.
- Glacier data download is an asynchronous process – once the data is successfully retrieved then SNS notification will be sent.
- Data retrieved from Glacier is copied to RRS – default retention period is 24 Hrs. but this can be changed depending upon the requirement (it can be extended or shorten)
- **RetrievalByteRange:** Instead of retrieving the complete archive data, one can request to download a portion of the data using **RetrievalByteRange** filed which needs to be set in the HTTP header while sending the request it should be in multiple of 100MB. In order to retrieve the correct ByteRange its advisable to maintain a repository outside glacier about the archive data.
- **Glacier Costing:**
 - No cost of moving data from EC2 instance to Glacier within a same availability zone.
 - Minimum charges of storing a data is 90 days.
 - While retrieving a data from glacier – along with the cost of data retrieval, temporary storage cost of RRS storage class will be applicable till the time retrieve data will be available.
- Glacier vault is a container where one can use to upload archive artifacts – the name of the vault must be unique per region (different region can have vault with same name). One can retrieve vault metadata or the inventory (retrieving vault inventory is an asynchronous process)..
- Glacier Vault Lock Policy: Glacier Vault Lock policy can be used to enforce regulatory and compliance need for archival data – once created the policy becomes immutable hence it can't be change. Example of the vault lock policy – WORM (Write Once Read Many). Vault Lock Policy can be use with access policies, e.g. Vault Lock policy for retaining the document for long period, and access policy to grant access to read vault inventory/metadata for a third-party provider user.
- Lifecycle policies
 - Lifecycle policies can be applicable to entire objects OR object with specific tag or prefix.
 - There are two types of lifecycle policies
 - **Transition Action:** After expire of the define period it moves the object from one storage class to other storage class.
 - **Expire Action:** After expire of the define period it deletes the object from s3 storage class.
 - Lifecycle policies can't be used to move an object from glacier to S3-Standard or S3-IA storage class.
- S3 encryption
 - Client-Side encryption: Client encrypts the data before uploading the same into S3 storage.
 - Server-Side encryption: Client uploads non-encrypted data, S3 service encrypts the data before storing it into S3 storage – while retrieving the data S3 service decrypts the data and send non-encrypted data back the client. Depending upon who manages the encryption key there are three type of SSE encryption available
 - **S3-SSE:** In this type of encryption the S3 service manages its own key.
 - **S3-SSE-KMS:** In this type of encryption the Key Management Service manages the key on behalf of S3
 - **S3-SSE-Client:** In this type of encryption the Key is managed by the client, and provide to S3 services for encryption. S3 doesn't stores the client key, in case the client losses the encryption key the data is lost.
 - S3-SSE how it works?
 - S3 service request for the data key to AWS KMS service

- KMS service generate the data-key and encrypts it with master key and send the encrypted data key and the plain-text data key back to the S3 service.
- S3 service encrypts the data using plain-text data key, and store the encrypted data key for future reference. Once data is encrypted it deletes the plain-text data key.
- For decryption, S3 sends the encrypted data key to the KMS service, which decrypts the encrypted data key using its master key and send back the plain-text data key back to the S3.
- S3 use the plain-text data key to decrypts the data and send it back to the client.
- S3 users AES (Advance Encryption Standard) 256-bit encryption for performing Server-Side Encryption.
- S3 rotates encryption keys periodically.
- There is NO extra charges for S3 server-side encryption.
- S3-SSE-Client Provide key
 - Client provide the encryption key to S3 service while uploading the data, S3 service user 256-AES encryption for encrypting the data.
 - Once the data is encrypted the client key is deleted.
 - While retrieving the data, client needs to provide the same encryption key which it has provided during uploading. If they key matches, S3 decrypts the data and make it available to the client.
- S3-KMS
 - First time, when a data is uploaded into S3-KMS it generates a Default Customer Master Key for that particular region. This key can be used for encrypting the data key which will be used for encrypting the actual object.
 - It's advisable to create own customer master key instead of using Default Customer Master Key, as it gives more flexibility to create, rotate, disable, define access control and audit the encryption key used for encrypting data.
- S3 bucket policy can be create in order to apply SSE for all uploaded object for that particular bucket.
- When uploading a request, one need to include **x-amz-server-side-encryption header** to enable SSE.
- Static-website hosting
 - S3 Static website hosting feature can be used to host static content from S3 bucket.
 - S3 static website hosting URL looks like
<http://<bucket-name>.s3-website-<AWS-Region>.amazonaws.com>
 - **S3 static website URL doesn't support SSL connection.**
 - S3 static website URL can be routed to custom domain URL using route53 CNAME.
 - S3 static website can route an incoming request based on the prefixes or the object name.
 - S3 hosted static website can redirect a request to the whole domain or pages within the domain or to a specific object.
 - There is no need to add ELB or autoscaling group to scale out the website, AWS s3 automatically scales as per the demand.
 - There is no additional cost of hosting static website.
 - S3 static website hosting doesn't allows requester pay request.

○ Difference between REST API and Static website

	Rest API	Static Website
Access control	Support both public and private content	Supports only public content
Error Message handling	Return back the HTTP base error code	Return back configured HTML error page.
Redirection Support	Not available	Can be redirect to object or to a bucket

	Rest API	Static Website
Request Support	Support all bucket operation	Supports ONLY GET and HEAD HTTP request
Response to GET and HEAD request	Return list of Object key hosted in the bucket	Return index.html page
SSL Support	Available	Not available

- Pre-Signed S3 URL: For sharing temporary access of a specific resources with the users how don't AWS account, one can use S3 URLs.
- Pre-Signed S3 URL can be generated by using SDKs, AWS explorer for visual studio, while creating a pre signed URL its mandatory to define an expiration date.
- Pre-Signed S3 URL can be used for both uploading and downloading object into S3 buckets.
- Cross Region Replication: This can be enabled at the bucket level to replicate the bucket object in different region. This is an automatic & asynchronous process managed by AWS on your behalf.
- Cross Region Replication can be made enable to the entire bucket or to object with specific key-name or object with specific prefix.
- Cross Region Replication can be configured with life cycle management rule
- Cross Region Replication is always 1-to-1 replication.
- During the replication the storage class can be changed, by default the storage class will be applied.
- Cross Region Replication happens over SSL channel
- To enable Cross Region Replication, it required to enable versioning.
- To enable Cross Region Replication, the source bucket owner should have access to the object & the object ACL in case bucket owner and the object owner are different. Source bucket owner should also have access to replicate object at the destination bucket. This can be granted by the destination bucket owner to the source bucket owner using bucket policy.
- During any change in the object, object ACL, object meta-data or during uploading / deleting object in the source bucket Cross Region Replication will be triggered.
- Any object that are existing in the bucket before Cross Region Replication is enabled will not be replicated, however any update to the existing object will be replicated.
- If an object is deleted from the source bucket Cross Region Replication will also attached the delete marker to the replicated bucket.
- If an object with specific version is deleted from the source bucket then the Cross-Region Replication WILL NOT add the delete marker to the replicated bucket (it will only delete the object version from the source bucket), this is done to prevent the malicious delete behavior.
- Objects that are encrypted using S3-SSE KMS and S3-SSR Customer Key WILL NOT be replicated as AWS will not have the encryption key to replicate the object at the destination bucket.
- Any sub-resources added to the bucket like life cycle policy or static website hosting will not be replicated to the destination bucket.
- Any object deleted by the life cycle policy WILL NOT be deleted at the destination bucket (but one can manually configure the same lifecycle policy at the destination bucket to overcome this).
- For Cross Region Replication following will be charged will be additional to the cost of the source bucket.
 - Uploading cost of the object at the destination bucket
 - Data transfer charges across region
 - Storage cost for the destination bucket
- Cross Region Origin: This can be enabled for the static website hosting to refer content from another domain (s3-bucket), by default this will not be enabled.
- Transfer Accelerator: TO improve upload performance, instead of uploading an object to the S3 region which can be far off from the uploader, use can upload the content into a nearby edge location from there the object will be transferred to the bucket over AWS infrastructure. [this doesn't guarantee performance enhancement, if performance is enhanced then this service is chargeable.]

- By default, transfer accelerator is not enabled, bucket owner can turn on this feature once turn-on it can take up to 30 minutes to enable transfer accelerator. Once enabled user can upload their content to transfer accelerator URL instead of S3 URL.

<https://<bucket-name>S3-accelerate.amazonaws.com>

- S3 Performance enhancements:
 - **Upload enhancements:** S3 internally maintain a list of indexes of the object that are store in S3 bucket, object with similar name are stored in same or nearby partition. Therefore, when object with similar prefix (sequential filename) are stored within S3 the performance degrades as its dependent on the IOPS of the storage partition. It's advisable to add random prefix to the object-name this way the object will be scattered across different portion within S3 storage resulting in better performance.
 - **Download enhancements:** To improve download performance it's advisable to configure CloudFront URLs. Users instead of downloading the content directly from the S3 bucket it will download from the CloudFront URL. Where data will be cached to improve performance, also network latency will be minimized as CloudFront service is distributed across regions.

PUT load (req/min)	GET load (req/min)	What needs to be done?
< 100	< 300	Noting
> 100	> 300	Add random prefix Add CloudFront distribution
>300	> 800	Open a support request with AWS, for preparing for workload

- **S3 Server Access logging:** AWS CloudTrail gives detailed API tracking for the S3 bucket level and object level operation, while s3 server access logs provides detailed visibility on the operation on the S3 objects – Both AWS CloudTrail and s3 server access logs can be use together to provide complete visibility on the S3 and its resources.

By default – s3 server access logs are disable, one can enable it at no additional logs, and configure the logs to be delivered to another S3 bucket within the same region of the on target s3 bucket.

Referrer and turnaround time of the s3 request are NOT available on the CloudTrail logs – for this information s3 server access logs need to be enable.

- S3 chargeable items
 - Storage charges /GB of data storage
 - Data transfer charges if applicable
 - No transfer charges within same region.
 - Data transfer to S3 is FREE
 - Data transfer to another region is chargeable
 - Data transfer to CloudFront is FREE.
 - Upload/Download request (GET/PUT request) per 1000 request
 - In case of **requesterPay** the upload/download (GET/PUT) request will be paid by the request along with the data transfer fees. When this feature is enabled it doesn't support anonymous access/BitTorrent access. [requesterPay option needs to be enable at the bucket level from AWS console]
 - Data Retrieval charges applicable to glacier and S3-IA
- S3 bucket event notification can be send to the following service, there is NO additional S3 charges for sending event notification. However, service charges will be applicable.
 - SNS
 - SQS
 - AWS Lambda function
- The following metrics can be recorded by **AWS CloudWatch**, filters (CloudWatch Dimensions) can be placed a the CloudWatch level to separate the CloudWatch data based on – bucket-name, storage-type, Prefix or Tags

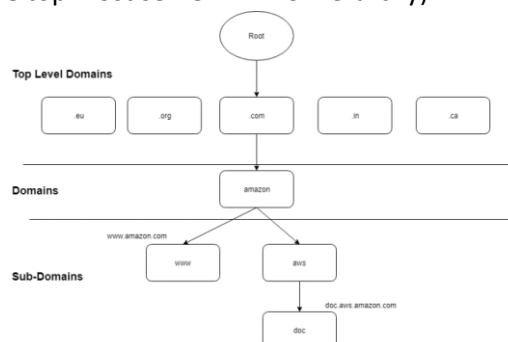
- S3 Request
- Bucket Storage
- Bucket Size
- All request
- HTTP 4XX and HTTP 5XX error responses
- Daily, bucket level CloudWatch metrics are enabled by default with no additional fee, however to have detailed monitoring (1 min) can be enabled at both Bucket Level and Object Level with additional charges. Up to 1000 metric configuration can be done at bucket level.
- By default, CloudTrail logs all API call at the bucket level, Object level tracking can be enabled. One needs to configure the CloudTrail to deliver logs to S3 bucket to reference later.

Announcement (Posted On: Jul 17, 2018): One can use logical or sequential naming patterns in S3 object naming without any performance implications. Amazon S3 now provides increased performance to support at least 3,500 requests per second to add data and 5,500 requests per second to retrieve data (i.e. 3,500 PUT/COPY/POST/DELETE and 5,500 GET/HEAD requests per second per prefix)

IMPORTANT NOTE: Correct way to upload objects into Glacier is using CLI or API. If the objects are uploaded to S3 then transition to Glacier (even if the object was stored momentarily in S3), the object will be managed by S3 even if the storage class will be Glacier. The Glacier feature like using Glacier Vault or setting up of retrieval option can't be implemented on those objects.

ROUTE 53

- AWS DNS (Domain Name System) is called Route 53
- Root-Server.org the root server (the top most server in DNS hierarchy)



- **DNS Zone:** this is the administrative authority that stores the address of all its registered domains.
- **Zone File contains:** This includes the DNS Zone configuration rule/mappings.
- **Primary Name Server:** This is the server that has a read-write copy of the specific zone. It has the highest authority on the zone.
- **Name server** are the ones that are responsible to answer the DNS query.
- Route 53 does the following
 - Register a domain for you
 - It's also a domain name registrar, it routes internet traffic to the domain resources.
 - It performs health checks on the registered resource and replies accordingly to a DNS query.
 - If the resources are found not healthy it can also send notification while routing the resources to the alternative resources.
 - Route 53 can be used in different ways
 - As domain registrar AND router for internet traffic to your resources
 - As router for internet traffic ONLY, while domain is registered with another domain registrar. [This is needed as Route 53 doesn't support all the TLD.]
 - When a domain is registered with Route 53, the following things happen. It creates its own DNS service for that domain
 - It creates a hosted zone, matching to the domain

- It allocates **four unique name servers** for each of the hosted zone, these name servers will be responsible for answering the DNS queries.
- It creates & maintains a link between the domain name/hosted zone and the name server allocated in SOA (*Start Of Authority*) file.

Note: Amazon Route 53 doesn't support all TLDs, if the required TLD is not available with the Route 53 one can register their domain with another domain registrar where the TLD is supported and then create a hosted zone for the same domain within Route 53 which will allocate the name server for that domain. Once Route53 allocates the name server, replace the domain registrar name servers with the route 53 name servers. Any internet traffic that come to the domain then will be served by the AWS name servers instead of the domain registrar name servers. This may take up to 48 hrs. to get reflected correctly based on the TTL values, as DNS servers cache name server information till TTL (time to live) counter is exhausted the DNS server will continue sending the old name servers.

- Inside the route 53 hosted zone, one needs to create record sets for delegation. Delegation means routing the internet traffic to specific name server.
- For registering a domain that is register with other domain registrar, one need to make sure the TLD (top level domain) does supported by Amazon AND, then one needs to get the authorization code from the current domain registrar.
- **Hosted Zone:** is a container where domain name will be mapped to name servers – this will help to route the internet traffic to the right host. It will have two entries by default – nameservers and SOA (Start Of The Authority). SOA will contain information about the hosted zone and the nameserver will have unique nameservers for the hosted zone.
- One can transfer a register domain from one AWS account to another AWS account, by creating a support ticket with AWS. However, connected hosted zone with the account will not be transferred as a part of this transfer. DNS will still continue to work, even though the hosted zone and domain register in different AWS account.
- Supported DNS type record
 - **A Record:** Address Record, it maps a hostname with the IP address
 - **AAAA Record:** IPv6 Address Record, it maps hostname with IPV6 address.
 - **CNAME Record:** Maps alias name to the Record, this are used to translate the actual domain name to its alias name. CNAME can't be configure for the top node domain, only for secondary domain CNAME can be configured. **For top node domain alias name can be configure instead of CNAME.** When CNAME is configure for subdomain, you can't create any other record for which the value is value of the CNAME record.
 - **NS Records:** Maps NS Servers with the Record
 - **SOA Record:** Start of the authority Record, every zone has one and only one SOA record. It contains
 - **Domain owner information:** it's usually an email ID
 - **The authoritative server:** the list of name server
 - **Serial Number:** incremental serial number which increments with the change in the zone data. This number is very important to sync up primary and secondary zone server.
 - **Refreshing time/Time To Live (TTL):** How long this cached value need to be shaved is decided by the TTL.
 - **MX Record:** Mail Exchange record, it defines where to deliver email for the users @ a domain name. **ONLY primary MX records are supported NOT the secondary MX records.**
 - **Alias Record (specific to Route 53 ONLY):** this is configured for DNS Route 53 record, to route DNS queries to the AWS services for which the IP address can change for services like (Classic Load balancer. Application Load Balancer, CloudFront, S3 bucket). Each time a query comes, Route 53 will resolve alias define in the alias record entry and respond to the DNS query with the IP addresses fetches. Unlike CNAME it can be configured for the top node also, it can point to other records on the hosted zone.
 - Records added to the hosted zone should have same prefix that of the hosted zone (domain name).

- NOTE: DNS quires done on the CNAME is chargeable wherein the quires done on the alias name is not chargeable.
- For Alias name one can't set TTL (time to live).

Difference Between CNAME and Alias Name record

CNAME	Alias
<p>CNAME can route a DNS query to any DNS Record, it does not need Route 53 as DNS service where the request is getting routed to.</p> <p>For Example, one can create a CNAME for mydomain.com that route its request to ABC.mydomain.com. OR it can route to mydomain.edu. It doesn't need Route 53 as DNS service for ABC.mydomain.com or mydomain.edu endpoints.</p>	<p>In case of alias name, you can only redirect DNS quires to selected AWS resources</p> <ul style="list-style-type: none"> - S3 bucket - CloudFront distribution - another record in the route 53 hosted zone that you are creating the alias record in. <p>For example, one can create an alias doc. mydomain.com that redirects the DNS quires to S3 bucket OR it can create an alias ABC. mydomain.com to route request to another record XYZ.mydomain.com is the same/different hosted zone.</p>
CNAME works only with the subdomain.	ALIAS can work for both domains and subdomains, but CNAME works only for subdomains.
One can't create CNAME for the top node a.k.a. zone apex.	<p>One can create alias name for top node as well for the sub domains.</p> <p>In most configurations, you can create an alias record that has the same name as the hosted zone (the zone apex). The one exception is when you want to redirect queries from the zone apex (such as example.com) to a record in the same hosted zone that has a type of CNAME (such as zenith.example.com). The alias record must have the same type as the record you're routing traffic to, and creating a CNAME record for the zone apex isn't supported even for an alias record.</p>
Route 53 charges for CNAME queries.	Route 53 doesn't charge for alias queries to AWS resources.
CNAME record redirects queries for a domain name regardless of record type.	Route 53 responds to a DNS query only when the name of the alias record (such as acme.example.com) and the type of the alias record (such as A or AAAA) match the name and type in the DNS query.
CNAME record can points to any DNS record hosted anywhere including the record that route 53 automatically creates when you create a policy record.	<p>Alias name record can only points to a cloudFront distribution, and ELB loadbalancer, an Amazon S3 bucket that is configure as static website OR to another record within the same Route 53 hosted zone where alias is created.</p> <p>One can't create an alias to that point to the record that route 53 automatically creates when you create a policy record.</p>
CNAME record is visible in the answer section of a reply from Route 53 DNS server.	Alias name is only visible in route 53 console or in route 53 API.

CNAME	Alias
CNAME record is followed by a recursive resolver.	Alias name is only followed inside Route 53; thus, alias record and the target must exist in Route 53.

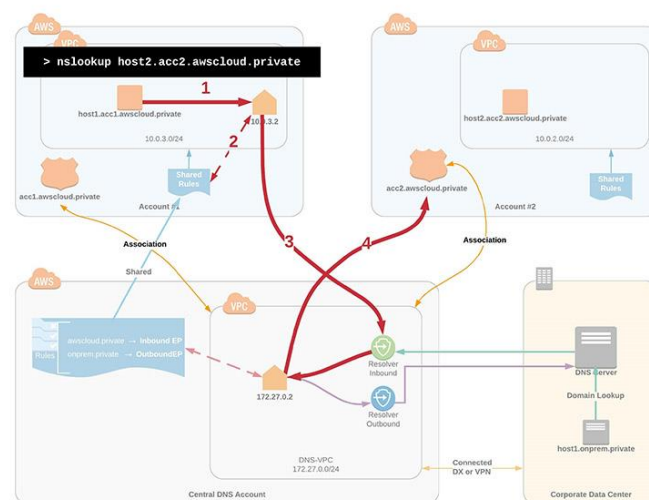
CNAME can't have any other record with same value. Thus, one can't have CNAME and alias name together.

- Route 53 routing policies
 - **Simple routing:** This is used to configure single resource for the domain example, a webserver.
 - **Failover routing:** This is used to configure active passive site, when the primary site is down queries will be automatically reply with the secondary site.
 - **Geolocation routing policy:** This is use when the routing needs to performed based on the geographic location of the requester. The content can be routed based on continent, country and based on united states. *In case of the geolocation routing, always the location which is more accurate will be preferred. Default routing policy needs to be added for those IPs which cannot be mapped or mapping is not available.*
 - **Latency routing policy:** based on the requester location, DNS query will be replied with the less latency site. *This is different from the geolocation routing as the routing is solely depends upon latency / performance, for some reason if the latency is less for resources outside the geographic location then the resource will be server by that resources instead of co-located resources.*
 - **Weighted routing policy:** use the routing incoming traffic based on the weighted routing rule configured within the policy.
 - **Geoproximity routing policy:** Based on the geo location of the requester and the resource AWS Route 53 will route the incoming request. One can optionally route more traffic or less traffic to a resource by specifying a value know as 'Bias'. By increasing or decreasing the **bias value** one can decide expand or shrink a region based on which it will decide where request needs to be routed to.
 - **Multivalued routing policy:** Multivalued routing allows returning more than one value (IP address) in response to a DNS query, which enable the users to choose IP from list of available IPs. **Up to 8 healthy endpoints can be return back in response to a single multi value routing policy request. If any un-healthy instance found, that endpoint will be excluded from the response. THIS IS FOR IMPLEMENTING CLIENT-SIDE LOAD BALANCING.**
- AWS Complex routing policy (nested routing policy)
 - When its desired to implement Latency based routing with weighted based routing. One needs to define weighted routing policy first then configure Latency based routing.
 - For implementing routing policy three things needs to be common among for each of the multiple webserver that is serving the same content. (For nested policy – at same level the Routing Policy/Name/Type needs to be common).
 - **Routing Policy** e.g. weighted
 - **Name** e.g. example.com
 - **Type** e.g. A Type
- AWS Route 53 Health Checks
 - This can be done on both alias base as wells as non-alias-based records.
 - For alias-based record one just need to enable to checkbox – enable health check.
 - For non-alias-based one need to define the heath check.
 - AWS Route53 health checker are distributed across the geographical location, if 18% of the heath checker demined a resource endpoint as un-healthy route53 will tag the resource endpoint as unhealthy.
- AWS Route 53 pricing
 - Per Hosted zone charges (there is no prorated charges, it will be charge for whole month)
 - Standard DNS query per million records

- Latency based DNS query per million record has different charges as AWS need to process more information.
 - Geoproximity and Geolocation base DNS query will be charge more than latency base DNS query and Standard DNS query.
 - For performing health check additional charges will be applicable, more charges will be applicable for the health check those are to done on premises.
 - Routing based on CNAME will be chargeable, routing based on Alias name is FREE.
- **DNS Resolver:** Resolving DNS queries between VPCs and on premises network: When a VPC is created, Route 53 Resolver automatically answers DNS queries for local VPC domain names for EC2 instances (ec2-192-0-2-44.compute-1.amazonaws.com) and records in private hosted zones (acme.example.com). For all other domain names, Resolver performs recursive lookups against public name servers.
- Reference link :** <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver.html>
- [Pending – to be added later]**

DNS resolver can manage DNS queries from/to on-premises domains. It can be used for the following use cases

- Resolving on-premises domains from workloads running in AWS.
 - AWS workload forwarding the request to its default DNS server connected to the VPC.
 - Forwarding rule shared by the central DNS server will be evaluated by the default DNS server.
 - Based on the configure rule the query will be forwarded to the on-premises DNS server.
- Resolving private domains in your AWS environment from workloads running on-premises.
 - On premises workload query on-premises DNS server.
 - Based on the conditional forwarding rule configure on the on-premises DNS server the DNS query will be forwarded to the VPC's default DNS server through DNS resolver inbound endpoint.
 - As default VPC is associated with the private hosted zone – Default VPC resolve the private domain and reply back to with its IP.
- Resolving private domains between workloads running in different AWS accounts.
 - The domain query is sent to the default DNS server for the VPC hosting source machine.
 - Because the VPC is associated with the shared forwarding rules, these rules will be evaluated.
 - A rule indicates that queries for awscloud.private zone should be forwarded to the resolver endpoint in DNS-VPC (for inbound endpoint IP addresses), which will then use the Amazon-provided default DNS to resolve the query.
 - Because DNS-VPC is associated with the acc2.awscloud.private hosted zone, the default DNS will use auto-defined rules to resolve this domain.



- **Traffic Flow:**

- Route 53 traffic flow provides a visual editor that help in creating complex trees easily
- The created configuration (routing tree) can be saved as a traffic policy
- One can associate the traffic policy with one or more domain names (such as example com) or subdomain names (such as www example com), in the same hosted zone or in multiple hosted zones.
- One can only use traffic flow to create configurations for public hosted zones

CLOUDFRONT

- AWS CloudFront, amazon managed content delivery service. AWS CloudFront leverages AWS edge location to distribute content there by reducing the distribution load on the origin server and improve customer experience as customer will be served from a nearby location instead of actual origin location. When customer make a request Route 53 redirect the request to the nearby edge location base on the latency , once the customer receives the request at the edge location, edge location verify if the request is available in its cache if not edge location makes a request to the origin server for the content over AWS infra structure (which is much faster then, downloading the content over internet). Once first few bytes of the request started to appearing edge location starts delivering the request to the requester. Once, the content is delivered to the requester the content will be stored in the edge location cache for severing future request.
- CloudFront performs better with Static Content as they can be cached easily. For Dynamic Content, depending on the query string content can be cached but still there will be a greater number of calls to the origin. Apart from getting caching benefits – AWS CloudFront also provide other benefits like
 - **Security to the content:** CloudFront seamlessly integrated with AWS WAF (Web Application Firewall) and AWS Shield to provide required protection from refine threats like DoSS (Denial of Service), it can also be integrated with ACM (Amazon Certificate Manager) service to manage and maintain secure connection to the content without overhead of managing/renewing certificates.
 - **Delivering content over vast network:** CloudFront provide content over vast network of **Edge-location** and **Reginal Edge-Location** spread all over the globe, ensure faster connectivity and lower latency from anywhere in the world.
 - **Greater Performance:** CloudFront Edge-location and Reginal-Edge-Location ensure content can be uploaded / downloaded faster will minimum latency as Edge-location and Reginal-Edge-Location are connected to the AWS infrastructure backend which provide high speed connectivity to the AWS services.
 - **Programable Content Delivery Network:** AWS Lambda@edge helps the lambda functions to run near requester location, ensure lower latency.
 - **Economical:** CloudFront charges only for the content that is transferred through its network, there is additional fee associated with the configuration or setup or transferring dynamic content. When the download rate is very high, it's advisable to use CloudFront as its cheaper to use CloudFront then directly accessing the content from the S3 bucket.
 - **Field level encryption:** Additional field level encryption can be turn on to encrypt sensitive data at the edge location closer to the requester and keep it encrypted till the application layer which has the password to decrypted the data for processing.
 - **Applying geo-restriction or geo-blocking:** Using CloudFront one can block specific object to be access from a particular geography. **This restriction is applicable at the distribution level.**
- **CloudFront is a global service.**
- CloudFront servers both ingress and egress connection i.e. for uploading content to the origin and for downloading content from the origin.
- There are two type of distribution that can be configure with the CloudFront
 - Web distribution
 - RTMP (Real Time Messaging Protocol) distribution.
- CloudFront can be configured using – AWS console, AWS API, AWS SDK, AWS CLI, AWS toll for power shell.
- Note: AWS doesn't recommend to use CloudFront for PCI DSS related information to store information in CloudFront Cache, however CloudFront is HIPPA (Health Insurance Portability and Accountability Act.) certified service. **CloudFront is NOT PCIDSS compliant.**

- CloudFront – **Edge Location** and **Regional Edge Location**. For less frequently access data, AWS automatically moves it to the regional Edge catch location which has more cache space than a regular Edge location. When requester request for a specific data, instead it fetch from the origin location, edge location can fetch the data from its nearby regional edge catch location adding performance enhancements to the customer experience. Regional Edge cache doesn't cache the dynamic content. Proxy method PUT/POST/PATCH/OPTION/DELETE doesn't go through regional cache.
- Default CloudFront cache for any object is 24 hours and minimum cache time is 0 hours, this can be set by configuring TTL (time-to-live) setting in CloudFront configuration.
- Custom s3 origin (static website) will only be served through CloudFront, regular S3 origin will not go through the CloudFront distribution.
- Web Distribution can be used to share the following over HTTP or HTTPS.
 - Static content like .js,.html,.css image files
 - Media content using on demand progressive download and APPLE HTTP live streaming.
 - Cannot be use for Adobe Flash Multimedia content over HTTP or HTTPS.
- RTMP distribution
 - Adobe Flash Multimedia can be share
- CloudFront web Distribution configuration setting
 - If the content is share with everyone or to restricted set of users.
 - Whether the CloudFront is distributing its content over HTTP or HTTPS
 - Whether CloudFront to forward cookies and/or query string to the origin, Whether to cache content base on all query parameters or on selected parameters.
 - Defining geo-restriction to the CloudFront URL – this will prevent content to distributed in the define geo-restriction locations. One can white list or black list countries.
 - Maximum limit of 200 web distribution per AWS account.
- CloudFront RTMP Distribution configuration setting
 - This distribution is required to send Adobe Flash Multimedia streaming videos.
 - For RTMP distribution the origin should be a S3 bucket unlike in case of web distribution where origin can be a webserver or a S3 bucket.
 - Maximum limit of 100 RTMP distribution per account.
- CloudFront Origin:
 - **Origin:** The DNS name of the S3 bucket for which CloudFront is to be configure to get objects from this origin.
 - **Origin Group:** When origin group is set, in an event of an origin failure (return specific HTTP code) the content will be severed to the request by forwarding the request to the secondary origin within the origin group. One can designate, primary origin and secondary origin in origin group.
 - **Custom Origin:** The DNS name of the HTTP server for which you want CloudFront to get object from this origin. The custom origin can server from an EC2 instance or from Elastic Load balancer or from S3 static website. For RTMP distribution, custom origin can't be used.

Few Guidelines for the custom origin

 - Use HTTP keep-Alive header to improve performance.
 - Ensure Date and Last modified header are accurate on the generated content
 - Don't use query String as those are not cached by CloudFront
 - Use Cache-Control header to made the content identifiable, whether it can be cached or not.
 - CloudFront accepts HTTP 1.1 request but makes only HTTP 1.0 request to backend origin server.
- CloudFront Configuration Steps (high level)
 - Define origin type - origin or custom origin
 - Define Access – Private or public access
 - Configure CloudFront distribution – web distribution or RTMP distribution
 - Define caching logic, and TTL.
- Cloud Front Alternative domain name: This can be done for the web-distribution or for RTMP-distribution.

- Create a DNS to route the alternative DNS to CloudFront domain name. This can be done by configure alias in Route 53 **OR for other DNS provides add a CNAME resources record set to the hosted zone for CNAME.**
- Path parameters the following things can be configure within the CloudFront distribution using a path parameter
 - The path patter e.g. /*.php or /*.jpg
 - For multiple origin, it can be used to configure CloudFront distribution to specify which request needs to be forwarded to which origin.
 - To configure, if the query parameters needs to be forwarded to the configured origin.
 - Whether, for accessing a specific file required a signed URL.
 - Allow HTTP or HTTPS for accessing a specific file.
 - The minimum time for which the file will remain in the CloudFront cache Regardless of the value define in the cache header which is set at the origin.
- Allowed CloudFront methods

Method Name	Description
GET	Request Data from a specific resource. Caching enabled.
POST	Submit a data to be processed to a specific resource.
HEAD	Same as that of the GET but returns only the HTTP header but not the document body. Caching enabled.
PUT	Upload a representation for the specific URI
DELETE	Delete a specific resource
OPTIONS	Return the HTTP method that server supports.
PATCH	Submit partial modification to the object

CloudFront ONLY cache's GET, HEAD method by default optionally it can be configured to cache OPTION method. However, it cannot be configured to cache PUT POST, DELETE and PATCH methods.

One can define the TTL is per object or at CloudFront level. Once the object is expired, on receiving the next call from the user, CloudFront forward the request to origin server – IF the CloudFront has the latest version of the object, origin server reply back with HTTP-304 status (Not Modified). IF CloudFront does not have the latest version of the object, it replies back with HTTP-200 along with the latest version of the object.

To be completed -

<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Expiration.html>

- Viewer Protocol policy: what protocol needs to be set between the requester and the edge location (CloudFront location)
 - **HTTP and HTTPS** – for requester to choose the HTTP or HTTPS.
 - **Redirect HTTP to HTTPS** – redirecting viewer HTTP call into HTTPS request.
 - **Only HTTPS** – requester needs to use ONLY HTTPS protocol.
- Origin Protocol policy:
 - HTTPS Only – the communication between CloudFront and the Origin will be HTTPS
 - Match Viewer Protocol – base on the selection of the requester protocol, communication between CloudFront and the Origin will be determined. This is mostly use in conjunction with Viewer Protocol Policy: Redirect HTTP to HTTPS or HTTPS only.

Note: for S3 static website origin – the origin policy is set to HTTP ONLY, as it doesn't support HTTPS.

[Pending] <https://aws.amazon.com/cloudfront/custom-ssl-domains/>

When alternative domain name is assigned to a CloudFront distribution, CloudFront will identify the correct certificate to apply by one of following ways

- **Use Server Name Indication (SNI) (Recommended)**

If CloudFront is configured to serve HTTPS requests using SNI, then it associates each of the alternative domain names with IP addresses from each of the edge locations. When a user initiates an HTTPS request with SNI in its request header, DNS routes the request to the IP address for the correct edge location. The IP address of the alternative name is determined during the SSL handshake. *The IP address isn't dedicated to a single client web distribution.*

- **By Using a dedicated IP address in each edge location:** If the client browsers do not support SNI, then a user can configure CloudFront to serve HTTPS requests using dedicated IP addresses. CloudFront associates an alternate domain name with a dedicated IP address in each CloudFront edge location. DNS routes the request to the IP address for a specific distribution in the applicable edge location. CloudFront uses the IP address to identify the distribution and to determine which SSL/TLS certificate to return to the viewer. [This solution works with all HTTPS requests regardless of the browser client used to view the request, however the origin owner needs to pay \$600 per custom certificate associated with one or more custom distributions.]
- **CloudFront protecting S3 origin from traffic spikes:** When a sudden traffic spike is observed and the CloudFront does not have the object in its cache to serve – it pauses briefly before serving the request, this helps in building the cache and the subsequent requests are served from the CloudFront cache instead of S3 origin.
- **CloudFront Object invalidation:** The process of refreshing the cache objects from the CloudFront web distribution is called Object Invalidation. This is performed at the object level and is chargeable. If a large number of objects within a same origin need to be refreshed then it's always advisable to build a new web distribution and replace the old one with the new one instead of invalidating each object as creation of a web distribution is not chargeable whereas object invalidation is.
- **Accessing Private Content through CloudFront Service.**
 - **Signed URL** – user needs to be authenticated themselves, once authenticated successfully, they will get their signed URL to access the private content. However, within the bucket we need to configure OAI (Origin Access Identity) within the object which are allowed so that they can only be accessed by CloudFront signed URL. (This can be done for both, origin and custom origin).
 - **Signed Cookies** – By inserting a signed cookie in the request header, one can allow accessing a private (restricted) content. With signed cookies the URL does not change; it remains the same.

Signed URL	Signed Cookies
When one needs to restrict access to a single item its advice to use signed URL over signed Cookies.	When one needs to restrict access to multiple files its advice to use signed cookies over signed URL.
For restricting RTMP distribution	Signed Cookies does not support RTMP distribution
When it's OK to change the URL	When it's not OK to change the URL
When client system doesn't accept cookies	When client system OK in accepting cookies.

Note : for signed URL one needs to define Origin Access Identity (OAI) this will remove the access of the S3 object over direct URL and only provide access through CloudFront URLs.

- **Video Streaming** – on-demand streaming & live streaming are two types of video streaming supported by AWS. Web distribution can be configured with one of the following configurations
 - Configuring on-demand with AWS Elemental Media Store
 - Configuring on-demand with Smooth Streaming
 - Configuring on-demand with progressive download
 - Configuring on-demand with Apple HTTP Live Streaming (HLS)

For configuring Adobe live streaming, one needs to use RTMP-based distribution. In case of the Adobe live streaming, there are two separate files that get distributed – media player and the Adobe media file content. For media player download one needs to use web distribution and for the Adobe media file content RTMP distribution needs to be downloaded. For RTMP distribution, the origin should be always a S3 bucket.

Note: On-demand streaming can be configured, but for live streaming one needs to involve AWS for configuration.

Signed Cookies are more preferred than Signed URLs in case of HLS encoding.

- **CloudFront Access Logs** – CloudFront access logs are captured by Access Logs. This needs to be enabled while configuring the distribution and can be redirected to store in a S3 bucket. CloudFront logs can be analyzed using Amazon Athena which is interactive query service.

All the trail logs to CloudFront API, made from AWS console, AWS SDK or AWS API are stored in CloudTrail API not in Access Logs.

NOTE: Once the CloudFront is enabled, the access logs from the CloudFront and the origin needs to be routed to the same Log Store (S3 bucket), to ensure that all the access logs are stored in a single place and audited as a whole.

- **CloudFront Pricing:** The following things are chargeable in CloudFront
 - Changes for S3 or EC2 instances depending upon the origin.
 - Data transfer to the requester are chargeable. (Data transfer, from the CloudFront and the requester is chargeable).
 - Data transfer charges applicable from uploading content from the CloudFront edge location to the origin will be chargeable.
 - HTTPS request
 - Request for Field Level encryption
 - Invalidation of the objects
 - No charges for data transfer between the CloudFront and the origin
 - No charges for regional CloudFront caching
 - No charges for AWS ACM TLS/SSL certificates for configuring SSL connection
 - No charges for shared CloudFront certificates
 - No extra charges for using HTTPS

For highly dynamic content – one can create a CloudFront distribution with enable query string forwarding and TTL set to 0 this will reduce the time it takes for TCP handshake to occur.

CLOUDTRAIL

- CloudTrail logs all management & data events of all users performed from AWS console, AWS API, AWS SDK or from AWS CLI.
- By default, CloudTrail logs all the events which can be view from CloudTrail console for 90 days, but it needs to be explicitly enable sending the logs to the specific bucket which can be use later, alternatively the logs can also be send to CloudWatch logs and CloudWatch events – which can be used to create custom metrics or CloudWatch alarm.
- Benefit of enabling CloudTrail it tracks - who make the change, when the change was made, what changes were made. It's needed for - For security & compliance needs, for tracking needs, for operational & troubleshooting needs.
- **[Important]** When enabling a CloudTrail there are option to enable it for a specific region or for all region. AWS recommend enabling it for all regions – so that all event from different region automatically gets log to the specified bucket.
- There are max 5 CloudTrail created per region. [Trail that are enable for all region counts in all region]
- Once enable, it takes 15 minutes to start logging. Once started it logs every 5 min.

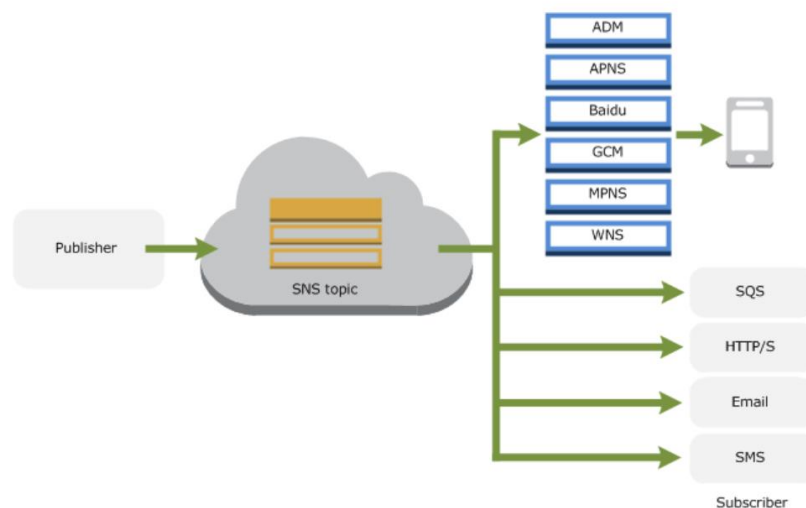
ENCRYPTION – KMS / (HSM) HARDWARE SECURITY MODULE

- The three things required for encryption are 1) **Data** that needs to be encrypted, 2) the **encryption algorithm**, and 3) the **encryption key**.

- There are two types of encryption key types – **Symmetric Encryption Key** (where encryption and decryption keys are same) OR **Asymmetric Encryption Key** (where encryption and decryption keys are different). **AWS supports Symmetric Encryption Key technique.**
- **KMI (Key Management Infrastructure) – Amazon KMS (Key Management Service)** is a type of KMI (Key Management Infrastructure) offered by Amazon for encryption key management. There are two parts to any KMI 1) **Storage layer** that protects the plaintext key(s) and 2) **Management layer** that authorized access to key usages.
- Amazon HSM (hardware security module) provides – **FIPS 140-2 Level 3** validated single tenant HSM cluster in your VPC for using and storing keys.
 - FIPS 140-2 Level 3 – (**Federal Information Processing Standards**) are set of standards that describes for processing documents, encryption algorithm and other information technology standards for use within non-military government agencies and the government contractors and vendors who works with the government agencies.
- Amazon HSM can be used for varieties of use cases
 - Digital Right Management (DRM)
 - Public Key Infrastructure
 - Document signing
 - Cryptographic functions
- Amazon KMS – Managed service by Amazon which is supported by FIPS 140-2 level 3 validated module (HSM). This can be leveraged by most of the Amazon services for their encryption and decryption needs. CloudTrail logs all the event related to the Amazon KMS – which helps in identifying how master key are getting use and by whom. ALSO logs the uses of the encryption keys for auditing, compliance and regulation needs.
- [important] KMS is a global service however the keys are confined to a region where they are created. (they cannot be transmitted outside the region where they have been created).
- KMS maintained multiple copy of the key across different availability zone across region to provide a durability of 99.999999999% of availability. User can import their own keys into KMS, in that case use needs to maintain their own key to re-imports the key if keys are lost due to KMS failure. KSM can't be use to export master key.
- Customer Master Key (CMK) can be used to generate, encrypt or decrypt data up-to 4kb (4086 bytes) – they are used to encrypt data key which then can be used to encrypt data of any size. [this is typically known as envelop encryption]. Customer Master Key (CMK) can't be exported outside of the KMS. Data key can be exported outside the KMS for data encryption.
- There are two types of CMK
 - **Customer Managed – CMK**
 - Customer needs to managed enabling and disabling of the CMK
 - Customer needs to rotate cryptographic material
 - Customer needs to create IAM policies to govern access to the CMK
 - Customer needs to use the CMK for their cryptographic operations, and can allow AWS service to use CMK on their behalf.
 - **AWS Managed – CMK**
 - AWS create, managed and use the CMK on behalf of the user
 - AWS keep the CMK unique for a particular region
 - ONLY those services that creates CMK uses AWS managed CMK for encryption purposes.
 - AWS managed CMK can be easily identifiable by their specific naming convention - aws/[service-name] e.g. aws/lambda or aws/rds etc.
- **Default Master Key**
 - When the first time an encrypted resource is created a default customer master key is created.
 - AWS manage the policies for the default customer master key to ensure new feature in supported service automatically.

SNS (SIMPLE NOTIFICATION SERVICE)

- AWS fully managed push notification service.
- Single message sent to all subscribers – Fan out message to all its subscriber immediately.
- Reliability of the SNS – AWS store SNS messages into 3 or more Availability Zone in a same region.
- Security of the SNS – AWS authenticate users before allowing access to the SNS topic. AWS also suggest use of secure channel (SSL) to connect to SNS topic over network to secure the message while transit.
- Authentication of the SNS topic – AWS requires all its publishers to signed the message with the secret key of the AWS ID & it validates the signature included in the request messages. By default, ONLY SNS owner can only publish messages to its SNS or give permission to other AWS account to publish messages to the SNS queue.
- SNS notification can be used to **push notification to the both mobile & desktop**. SNS Mobile push notification service supports on the following platform.
 - Apple Devices
 - Google Devices
 - Fire OS
 - Windows device
 - Android Devices over china through Baidu Cloud Service.
- To received SNS push notification mobile devices should have the APP installed on their application, in case of Apple, Android, Fire OS platform it also required to provide explicit permission on the device to received push notification.
- Currently, amazon push messaging supports following platforms
 - Amazon Device Messaging Service (ADM)
 - Apple Push notification service (APNS)
 - Google Cloud Messaging (GCM) – for windows 8+ and 8.1+ phones
 - Windows push notification service (WPNS) – for windows 8+ and 8.1+ phones
 - Microsoft Push Notification Service (MPNS) – for windows 7+ phones
 - Baidu Cloud Service for pushing notification to android device in China



- Up to 10 message attributes can be pass along with the message body (payload) which can be used or message filtering. Attribute can carry only string values not JSON object.
- SNS direct messaging – send specific messages to a single endpoint there by allowing to deliver message directly to an intended endpoint.
- CloudTrail logs SNS request details, which can be audited later – however only authenticated API calls are ONLY logged in the CloudTrail.

BEANSTALK

- Amazon BeanStalk provide easy way to deploying application as it manages all the underline capacity provisioning, complexity of instances, load balancing, application health monitoring, application logging for hosting application in its supported language.
- To use Elastic Beanstalk, create an application, upload an application version in the form of an application source bundle (for example, a Java .war file) to Elastic Beanstalk, and then provide some information about the application. Elastic Beanstalk automatically launches an environment and creates and configures the AWS resources needed to run your code. After your environment is launched, you can then manage your environment and deploy new application versions.
- Supporting subset of platform for deploying applications
 - Supported languages: JAVA, .NET, Node.js, Python, Ruby, GO
 - Supported platforms: Web container, Docker container

CloudFormation	OpsWorks	BeanStalk
Need control more on the infrastructure	Need more control over deploying application	Less control over deploying application
More towards – configuring infrastructure.	Give more control over deploying application	Less control, less complexity for application deployment.
		In case of BeanStalk application health monitoring is included.
Stack	Stack	Environment

- There is no additional charge for Elastic Beanstalk. You pay only for the underlying AWS resources that your application consumes.
- Elastic beanstalk can also be used to build custom platform, one can create their own custom build platform and use it like other build platform available to them.
- Instances running on Amazon Beanstalk doesn't have persistence local storage. When an instance is terminated the data stored in the EC2 instance will be lost. (RDS instance created a within amazon beanstalk will lose its stored data once the instance is terminated).
- Elastic Beanstalk can use other persistence store to store its data like – S3, RDS, dynamoDB, RDS, EFS.
- If the underline aws resources created by Elastic beanstalk are terminated or modified directly then there is a possibility that the full environment becomes unusable – rebuilding the environment will terminate existing resources and recreate the new resources with same configuration.
- Within 6 weeks (42 days) of a termination of an Elastic Beanstalk environment – it can be rebuilt. When rebuild, aws recreate all the environment with the same name and configuration (if available with the same ID).
- **Application** – In case of Amazon Beanstalk, application is the container that run on the amazon beanstalk environment. It comprises of – versions of source code, saved configuration logs and other artifacts. Amazon beanstalk application is logical collection of – environment, versions & configuration. Its conceptually equivalent to a folder. Deleting an application, deletes all the source code version, saved configuration and the environment.
- **Application version** – refers to a unique, specific labeled, iterable version of a deployable code, which points to a specific deployable file in a S3 bucket within Amazon beanstalk environment. Application can have multiple version within a same Amazon beanstalk environment, but only one running version.
- **Environment** – the container that run a specific application version. Multiple application version of a same application can be run simultaneously in different environment.
- **Environment Tier** – User selects environment tier, based on the environment tier value amazon beanstalk provision resources to support the application. Web application runs web environment, backend processing application runs on worker application.
- **Environment Configuration** – These are the collection of parameters and the settings based on which an environment behaves to cater the needs of an application. Any change in the environment configuration – amazon beanstalk will apply the changes to the underline AWS resources on your behalf, else it will terminate the instances and create a new resource with the new configuration.

- **Saved Configuration Template:** Starting point for creating unique configuration for the amazon beanstalk environment. Configuration template can be update using beanstalk console, AWS CLI or through API. When using saved configuration within CLI or through API they are referred as Template.
- **Platform:** It's a combination of Operating System, programming language runtime, application server and elastic beanstalk components. Beanstalk provides various different platforms; user needs to create their application targeting a specific platform. It provides platforms for different programming language, application servers and docker container.
- **Elastic bean cloud Shared Responsibility Model** – It applies the patches and mirror release on your behalf. Update major releases as available platform within 30 days of the releases.

Customer Responsibility:

- Implementing application level security or any data or components that was downloaded separately and not available as part of the platform.
- Keep platform updated, migrating retired platform to supported platform
- Act on failed update notification, *(when an update failed, amazon beanstalk notifies the customers, for them to act on it)*
- Patching OS in case customer opt out from Elastic Beanstalk to patch platform.
- Managing security for the AWS services that are hosted outside the beanstalk platform.

AWS SERVICE – APPLICATION LOAD BALANCER

- There are three types of elastic load balancers – *Application Load balancer*, *Classic Load Balancer* and *Network Load Balancer*.

	Classic Load Balancer	Application Load Balancer	Network Load Balancer
Layer 4	NO	NO	YES
Layer 7	YES	YES	NO

- Classic Load Balancer can have up to 100 listeners. Each will have a 1:1 static mapping between frontend and backend listeners.
- Limitation of a classic load balancer
 - In case of the classic load balancer , there is NO way to manage separate feet of EC2 instnaces for separte application endpoints using a single classic load balancer.
 - CLB cannot perform health checks on the ECS container level, healthcheck are perform in the EC2 instance level where one or more container can be started. With the introduction of the target group in Application Load Balancer this problem has been overcome in Application Load Balancer. Similar services can be group together in a single target group.
- There can be multiple application (target group) configure within a single application Load Balancer however it's not recommended to group all application into a single Load Balancer. It's always advisable to split the application and maintain a lower count.
- Application Load Balancer supports – layer 7 protocols HTTP, HTTPS, HTTP/2 and WebSocket.
- Components of Application Load Balancers
 - **SNI** connected targets within its target groups.
 - **Listeners:** listens to the HTTP and HTTPS connection from the client & forward the request to the Target Group based on the rules define.
 - **Target Groups:** It's a regional construct, a logical grouping of targets associated with a single load balancer. Auto scaling services scale each target groups individually. For each target group there is a specific protocol and a target. Application Load Balancer can route incoming request to multiple target groups. One cannot mix and match target groups – they should be of similar types with same protocol & port.

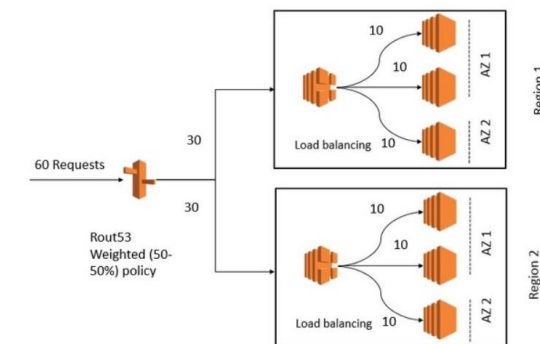
- **Target:** These are the endpoints specifies in the target group. They can be EC2 instances, Lambda functions, Application running on an ECS container. **[Internet routable endpoints cannot be a target group].**

When to use ip addresses to register targets within a target group?

For instances in peered VPC, AWS resources that are reference by IP address like database services, on-premises resources through direct connect or vpn connection.

- **Rules (Condition, Actions, Priority):** Up to 100 rules can be define, lowest priority value to highest priority value. Default will be the last rule to evaluate. Each rule can have a condition each condition has a host and a path (optional). Forwarded action can ONLY be create within a rule.
- **Content base routing:** it forwards the request to a specific target group
 - Path base routing - domain.com/service & domain.com/service2
 - Host bath routing – domain.com / content.domain.com
- **Dynamic host port mapping** – task running on a container services needs to map task port to the host port, allowing multiple task from the same service per container. ECS will automatically register task with the ALB using dynamic-host-port mapping. **When the task is launched, the container is register with the application load balancer as an instance and a port combination, AND the traffic will be routed to the combination of the instance and port. This allows one to have multiple tasks from a single service on a same container instance with same port.**
- **There are four type of monitoring possible with ALB**
 - **CloudWatch** – Every 60 second if there is a traffic flowing through the load balancer.
 - **CloudTrail** – Logs all calls (no cost of enabling, storage cost applicable)
 - **Access Logs** – contains more details of the calls made to the ALB (no cost of enabling only storage cost applicable). Does not guaranties all request logging.
 - **Request Tracing** – ALB allows request tracing by adding a *x-Amzn-Trace-Id* header from client to target.

Load balancing across region: Load balancer is a regional construct – it can ONLY load balance traffic within different AZ in the same region, in order to get load balanced across region one need to have Route 53 place in front of regional load balancers.



AWS SERVICE – NETWORK LOAD BALANCER

- Network load balancer function on the Layer 4 of the OSI model, it can handle millions of requests per minute. Once a connection is established network load balancer route request to the target group attached to it.
- For TCP connection NLB route the traffic based on the flow hash algorithm which is based on source IP + source port + destination ip + destination port + sequence number: Since source IP, Source ports & sequence number are changing NLB route request to different target endpoints.
- In case of UDP NLB route the traffic based on the flow hash algorithm which is based on source IP + source port + destination ip + destination port. Since source IP and source port are constant it routes traffic to same target unlike in case TCP connections.
- NLP receives static IP for each availability zone it enables for, if it's an internet facing NLB then there is an option to attach an elastic IP address to it. Targets can be attached to an NLB based on Instance ID or IP addresses.

[NEW: Network load balancer now supports resource and tag base permission, enabling users to have finer grain IAM policies to control action on network Loadbalancer. Prior to this ONLY resource based IAM policies can be applied – like IAM policies to allow access to DeleteLoadBalancer API can delete any Loadbalancer in the account, but now the access can be restricted to a specific ARN only.]

NOTE: Network Loadbalancer now supported – load balancing between UDP traffic to deploy connectionless services for online gaming, IoT, streaming, media transfer, and native UDP applications. UDP target currently don't support – IPs and Private links.

Network load balancer features to handle – millions of requests per second.

Static IP addresses: Each of the network Loadbalancer instance will be allocated static IP addresses – one per AZ. Alternatively, one can also assigned elastic IP to NLB instance in each AZs for full control. This can be use in situation where IP address needs to be hardcoded into DNS entry or in firewall for white listing. The IP per AZ feature reduces latency with improve performance, improve availability through isolation and fault tolerance at the same time increase transparency on the operation of NLB to the client application.

Source IP Address Prevention: In network Loadbalancer (if selected instance ID as target) the original source ip address and ports remains unmodified, there by reducing the need to support for x-forwarded for and proxy protocol. Since the source ip address and ports remains unmodified, targets response back direct to the source NOT through the NLB, hence the targets need to have access to internet through a NAT instance/gateway or an IGW.

If selected Instance IP as target, in order to get the source IP one needs to configure Proxy Protocol.

Proxy Protocol: NLB support Proxy Protocol V2 – its configure at the target level and disable by default. This is required, for the application that needs to know the source ip addresses.

Supports long running connections: NLB build in fault tolerant support long running connections (connections that are open for month and years) this makes it ideal for IoT, gamming and messaging applications.

Failover – Built in support for Route53 health check – NLB support failover between IP addresses within and across region. **NLB supports cross region VPC support – it can reach out to IP based target in peered VPC across different AWS region.**

Cross region loads balancing: Network load balancer distribute incoming traffic load equally regardless of the availability zone.

Network load balancer configuration – using elastic IP addresses

Step 1: Configure Load Balancer

Basic Configuration

To configure your load balancer, provide a name, select a scheme, specify one or more listeners, and select a network. The default configuration is an Internet-facing load balancer in the selected network with a listener that receives TCP traffic on port 80.

Name

Scheme ☒ internet-facing ☐ internal

Listeners

A listener is a process that checks for connection requests, using the protocol and port that you configured.

Load Balancer Protocol	Load Balancer Port
TCP	80

Availability Zones

Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availability Zones only. You can specify only one subnet per Availability Zone. You may also add one Elastic IP per Availability Zone if you wish to have specific addresses for your load balancer.

[Click here to manage your Elastic IPs.](#)

VPC

Availability Zone	Subnet ID	Subnet IPv4 CIDR	Elastic IP	Name
<input checked="" type="checkbox"/> us-west-2a	subnet-d574b5a2	172.16.0.0/24	<input type="text" value="34.214.96.77"/>	
<input checked="" type="checkbox"/> us-west-2c	subnet-3e27c667	172.16.1.0/24	<input type="text" value="52.33.71.127"/>	

Tags

Apply tags to your load balancer to help organize and identify them.

Key	Value
app	web-app

Health check for Network Loadbalancer

There are two types of health check available on Network Loadbalancer – Active health check, Passive health checks.

Active health checks are the one that one can configure on the NLB, which periodically sends request to the targets to learn about their health.

Passive health check is not configured – they are derived by observing the connection response received from the target(s). [This passive health check cannot be disabled.]

NOTE: if there is no healthy instance to route the traffic – NLB forwards the traffic to all the targets and let target response to the traffic. IT does not drop the traffic.

AWS SERVICE – ELB COMPARISON

- Comparison of three Elastic Load Balancers.

Ref Link : <https://aws.amazon.com/elasticloadbalancing/features/#compare>

Feature	Application Load Balancer	Network Load Balancer	Classic Load Balancer
Protocol	HTTP/HTTPS /HTTP2/Websocket	UDP/TSL/TCP	HTTP/HTTPS TCP, SSL/TSL
Platforms	ONLY VPC	ONLY VPC	VPC, Classic EC2
Health check	YES	YES	YES
CloudWatch	Additional Metrics	Additional Metrics	Basic Metrics

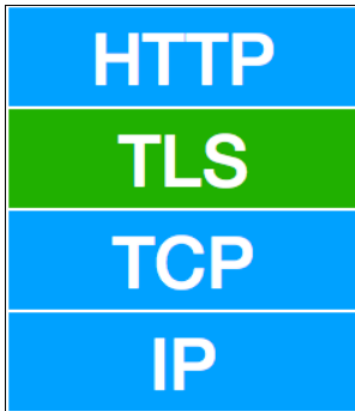
Feature	Application Load Balancer	Network Load Balancer	Classic Load Balancer
CloudTrail	Adding log details	Adding log details	Basic logging
Zonal Failover	NO	NO	NO
Connection draining (deregistration delay)	YES	YES	YES
Load Balancing to multiple ports on the same instance	YES	YES	NO
IP addresses as targets	YES	YES For TLS/TCP	NO
Load balancer delete protection	YES	YES	NO
Configurable idle connection timeout	YES	NO	YES
Cross-Zone load balancing	Enable by default	Enable by default	Need to be enabled
Sticky sessions	YES	NO	YES
Static IP	NO	YES	NO
Elastic IP	NO	YES	NO
Preserver Source Address	NO	YES	NO
Resource-based IAM Permissions	YES	YES	YES
Tag-based IAM Permission	YES	YES	NO
Slow-Start	YES	NO	NO
WebSocket	YES	YES	NO
Private Link Support	NO	YES (TCP,TLS)	
Request Tracing	YES	NO	NO
Source IP address CIDR-based routing	YES	NO	NO
Layer 7 feature ONLY supported in ALB			
Path-Based Routing	YES	NO	NO
Host-Based Routing	YES	NO	NO
Native HTTP/2	YES	NO	NO
Redirects	YES	NO	NO
Fixed response	YES	NO	NO
Lambda functions as targets	YES	NO	NO
HTTP header-based routing	YES	NO	NO
HTTP method-based routing	YES	NO	NO
Query string parameter-based routing	YES	NO	NO
Security Configuration			
SSL Off-loading	YES	YES	YES
Server Name Indication (SNI)	YES		
Back-end server encryption	YES	YES	YES
Back-end server authentication			YES
User authentication	YES		
Custom Security Policy			YES

Sticky session – helps to route the same client request to same target endpoint if client supports cookies. ALB uses cookie name **AWSALB** for tracking session, content of this cookies is encrypted.

Delayed de-registration helping inflight request to be completed before de registration.

Idea time out – once the idea timeout elapse, the ALB close the front-end connection. For backend, its suggested to use keepalive option to ensure connection are not terminated.

Supports SNI server name identification, which helps in binding multiple certificates to the same secure listener on the ALB. Server Name Indication (SNI) is an extension to the Transport Layer Security (TLS) computer networking protocol by which a client indicates which hostname it is attempting to connect to at the start of the handshaking process. Classic Loadbalancer does not support SNIS certificates. **Application Load Balancers Now Support Multiple TLS Certificates With Smart Selection Using SNI** : <https://aws.amazon.com/blogs/aws/new-application-load-balancer-sni/>



SSL and TSL are not interchangeable terms – TSL is applicable to the Layer 4 (network layer), while HTTP is applicable to Layer 7 (application layer).

TLS is a protocol for securely transmitting data like passwords, cookies, and credit card numbers. It enables privacy, authentication, and integrity of the data being transmitted. TLS uses certificate-based authentication where certificates are like ID cards for the websites. HTTPS is basically HTTP connection which is delivering the data secured using SSL/TLS.

With SNI support we're making it easy to use more than one certificate with the same ALB. The most common reason you might want to use multiple certificates is to handle different domains with the same load balancer. It's always been possible to use wildcard and subject-alternate-name (SAN) certificates with ALB, but these come with limitations. Wildcard certificates only work for related subdomains that match a simple pattern and while SAN certificates can support many different domains, the same certificate authority has to authenticate each one. That means one has to reauthenticate and reprovision your certificate everytime you add a new domain.

ALB Access Logs now include the client's requested hostname and the certificate ARN used. If the "hostname" field is empty (represented by a "-") the client did not use the SNI extension in their request.

Certificates can be stored in ACM or IAM. The certificates created by ACM are of 4096 key length and EC certificates. One cannot use the certificates directly on the ELB – alternatively one can create and get CA Signed uploaded to ACM which can be used in ELB.

ALB allows binding multiple certificates for the same domain(s) to a secure listener. Your ALB will choose the optimal certificate based on following factors including the capabilities of the client. If no SNI is used, ALB selects the default certificate (defined during ALB creation). One can bind up to 25 certificates per load balancer (not counting the default certificate).

- Public key algorithm (prefer ECDSA over RSA)
- Hashing algorithm (prefer SHA over MD5)
- Key length (prefer the largest)
- Validity period.

Supports IPv4 and IPv6 on the front end – backed ONLY support IPv4. For internal ALB ONLY supports IPv4

Fail open – If there is no healthy instance in any of the availability zone, ALB doesn't drop the request instead it sends the request to all the targets hoping one of the instances to response.

Slow-Start: ALB supports slow start feature, where newly added target does not overwhelm with full load from the start instead it slowly gets its fair share of the load.

AWS SERVICE – ELASTIC CACHE

- Elastic Cache is an AWS managed in memory key-value store – improve performance by storing frequently access data into in-memory cache thereby offloading frequently multiple read queries from the data bases.
- Elastic Cache EC2 instance are of type reserved instance or on-demand instance type but not of Spot instance type.
- EC2 instance deployed for hosting Elastic Cache can't be accessible from internet or from different VPC.
- For hosting Elastic Cache EC2 instance one need to configure the subnet – where elastic cache cluster will be hosted. Currently changing of the subnet group is NOT supported.
- If the Elastic cache node fails AWS, will automatically replace the same with another node.
- Elastic Cache Cluster can be of a single node in one subnet or comprises of multiple node span across multiple subnet from the same subnet group.
- Its always advisable to access the Elastic Cache cluster using endpoint rather than using it through IP addresses.
- Elastic cache has two type of engine
 - **Memcached** – This is a pure cache, it cannot be use as database. Its not a persistence store. If a single node within the cluster is fails the entire data stored in the cache is lost.
Max of 100 nodes of Memcached instances can be create per region, in a single cluster 1-20 nodes can be added. (soft-limit)
This is ideally suited for
 - Caching data from RDS – SQL and noSQL databases
 - Caching data for dynamically generated webpages
 - Transition session data
 - High frequency counter for admission control for high volume web-app.
 - Memcached can be integrated to SNS, to notify on any node failure/recovery.
 - It supports auto-discovery of the new node added/removed from the cluster.
 - It can be *vertically scale-in/scale-out* (by adding/removing more nodes to/from the cluster)
 - It can be *horizontally scale-in/scale-out* (by upgrading/downgrading the EC2 node instance family)
 - Since Memcached is a non-persistence cache, scaling in/scaling-out would need a new cluster to be created. Data of the old cluster will be lost, and the data needs to add freshly to the newly added cluster.
 - Memcached DOESNOT supports Multi-AZ failover, replication, snapshots for backup/restore, any failure to the node resulting in cache data loss.
 - To minimizing the impact of the node loss, one can distribute the node across multiple AZ, and the application needs to spread-out the data stored in the cache data across different node in different AZ.
 - **Redis** – Redis can also be use as in memory database.
 - Fastest in memory NoSQL database which can be use as cache store, it supports replication and snapshotting. The snapshot can be stored in S3 bucket which can be use to recover the cache data into a new cluster node. Automatic backup of the Redis cache can be enable, also its possible to manually backed up the cache. [When the Redis cache is deleted, the automatic backups are deleted however the manual snapshot does not get deleted].
 - Redis does support multi-AZ deployment, the read-replicas can be created in multiple AZ within the same region.
 - Redis support cluster node disable -in this mode ONLY one shard can be use which has one Read/Write primary node and 0-5 replication node, however this replication node can be spread across multiple different availability zone.
 - **Redis supports clustering mode – in this mode there can be up to 15 shards in a single cluster, with each shard can have up to 5 read replicas of their own.**
 - When a Redis detect a failure of a primary node, then it automatically promotes one of the read-replicas which is having least lag with the primary will be promoted as primary node, and the remaining read replicas started syncing up with that node. DNS records will be automatically updated with the new primary node IP addresses.

- There is no direct way to move the Redis cache to a different region, however there is a workaround available where one can export a snapshot into S3 bucket and then copy it another region of choice and create a new Redis cache from the backup.
- It supports master slave replication for high availability. The data from the primary Redis node will be asynchronously sync up with the read-replica node.
- Redis cache supports multi-A failover.
- Its idea for storing data for WEB, WEB APP, MOBILE APP, GAMMING Leader Board, Ad-Tech etc.
- Caching Strategy
 - Lazy loading – First time when a new record is requested the application reads the data from the data-store and update the cache. The subsequent read requests are then served from the cache store instead of main data-store.
 - Write Through -
 - TTL – setting time to live, at the request level.

AWS SERVICE – API GATEWAY

- API gateway supports (stateless)REST and (state full) web socket-based APIs.
- Supports powerful and flexible authentication mechanism - AWS IAM, *AWS Lambda Authorizer function* and AWS Cognito user pool.
- AWS X-Ray can be used to trace and monitor API calls.
- API Gateway, is amazon managed service which helps in hosting client API where it performs most of the heavy lifting on behalf of the client.
- **API Gateway Pricing** – there is no upfront cost associated with the API gateway. The API costing involves number of transactions made through the API gateway and amount of data that passes through the API gateway.
- **What is a Resource?** A resource is an object with a type, associated data, relationship with other objects, and a set of method that operates it.
- **API Gateway** is a collection of the resources and method that are integrated with the backend HTTP endpoint, AWS lambda function & other AWS services.
- **There are 4 (four) types of integration possible with the API gateway**
 - **AWS Service:** Request received at the AWS front end will be pass the backend after processing the request within the API gateway.
 - **AWS Proxy:** Pass the request directly to the backend service
 - **HTTP:** Pass the HTTP request directly to the backend service
 - **HTTP Proxy:** HTTP request received at the AWS front end will be pass the backend after processing the request within the API gateway.
- **API resource supports one or more standard HTTP method like (GET, POST, PUT, DELETE, HEAD), it also supports ANY method in its end point.**
- **API gateway only supports – HTTPS endpoints. It doesn't support un encrypted HTTP end point.**
- API Gateway can be configured to use custom domain, in such customer needs to create a route 53 alias name for the API gateway default domain and also need to manage the certificates for the custom domain.
- Benefits of API gateway
 - Provides a robust, scalable, secure access to the backend hosted API.
 - Supports multiple version of same API and also supports different stages of the API (Like DEV, TEST, PROD).
 - Supports creation and distribution of the API key for the developers.
 - Use of AWS Sig-v4 for the authorize access to the API.
 - API response can be cached at API gateway cache – this is a chargeable service. API gateway cache can be enabled for a specific level, one can define their own TTL for the cache value. API gateway management API provides a way to invalidate cache for a specific stage.

- Throttle and Monitor requests to prevent your backend. *(HTTP 249 will be sent back to the requester if the request exceeded throttling limits - throttling limits can be set as granular level as throttling limit for GET/PUT method level)
- Can take advantage of the reduce latency and DoSS attack by exposing the API gateway through CloudFront URL.
- Supports swagger.
- Can transform inbound request as per the mention templates
- CORS –Cross origin resource sharing is a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served. API gateway supports CORS, it needs to be enabled at method level.
- **Provide API gateway dashboard.**
- Controlling access to backend REST API, through API gateway
 - Resource based Policies can be granted to
 - IAM User/IAM Role/Service Role for a same account OR from cross account access.
 - Specific CIDR range – can be whitelisted or backlisted.
 - To a VPC or VPC endpoint.
 - Standard IAM role and policies : Can be add granular level access control using IAM policies which can than grant users to perform action on the AWS gateway.
 - Cross-origin resource sharing (CORS)
 - Lambda Authorizer
 - AWS Cognito user pool.
 - Client-side SSL certificates: API Gateway can be loaded with the SSL certificate, which can help the backend REST services to authorized it before responding to its request.
 - Track and limit, backend API usages: API keys can be assigned to each client systems their usages can be tracked and if required limit using based on the API key present in the request.
- AWS Gateway authorizers
 - Using Lambda Authorizer (customize authentication).
 - Using Cognito pool user:
- AWS Gateway request tracking
 - X-Ray tracks end-to-end request – it work on simple three-tier application as well as for complex microservice interaction. One can use X-Rays to analyzing time taking in each stage.

AWS SERVICE – AWS LAMBDA

- AWS compute service without any need for provisioning and maintaining servers.
- AWS Lambda functions are **Regional Service**.
- AWS Lambda supports following languages
 - Node.js
 - Java (java 8 compatible)
 - Ruby
 - Python
 - C#
 - Go Lang
- Components of Lambda function
 - **Function:** It's the script (code) that runs on the Lambda runtime to process event into response.
 - **Runtime:** It helps in executing different languages to lambda functions. It sits between the Function and the lambda service.
 - **Layer:** it's a distribution mechanism for libraries, custom runtimes, and other dependencies that are required by the functions. Its best practice to keep the decencies in the layer to keep the deployment package size smaller.
 - **Event Source:** This triggers the Lambda function.

- **Downstream Resources:** This is the AWS service that lambda function invoke post completion of the lambda function.
- **Log Streams:** though the lambda functions are monitored automatically by the CloudWatch, one can also annotated the code to log custom logs statements which can be viewed to analysed the code execution.
- AWS lambda function triggers
 - Based Events (through event source mapping specific Lambda function can be invoked).
 - By API gateway request
 - Through API call make form SDK
 - Based on the input stream – DynamoDB and Kinesis are stream based services for these services the event will be configure in the AWS lambda side, instead of in the event generator side like in the case of S3.
- The following service can be used for triggering AWS Lambda function
 - Amazon S3
 - Amazon Dynamo DB
 - Amazon kinesis streams
 - Amazon Simple Notification Service
 - Amazon Simple Email Service
 - Amazon Cognito
 - **AWS CloudFormation**
 - AWS CloudWatch logs
 - AWS CloudWatch Events
 - AWS CloudComits
 - Scheduled Events (powered by AWS CloudWatch events)
 - AWS Config
 - AWS Alexa
 - AWS Lex
 - AWS API Gateway
 - AWS IoT Button
 - AWS CloudFront
 - AWS Kinesis Firehouse
 - Other AWS Event sources
- Aws Lambda Scaling
 - **AWS lambda dynamically scales up to meet the demand for increasing traffic, till account specific concurrent limit is reached. i.e. 1000 concurrent transactions per account.**
 - To cater burst of concurrent request, AWS lambda will scale up automatically by a pre-define amount depending upon the region.
 - Lambda function depends on AWS EC2 instances for providing elastic network interfaces for VPC enable lambda function, these functions are also dependent on EC2 scaling limits as they scale up.
 - AWS Lambda Scaling for Stream base trigger – depending upon the number of shards configure number of concurrent lambda function will be triggered.
 - AWS Lambda Scaling for Non-Stream base trigger – it depending on the number of concurrent message, the lambda function will seamlessly scale up to the max number of concurrent event received, or maximum invocation limit is reached whichever is lower.
 - Maximum concurrent transaction per account is set to 1000 concurrent transactions – this is a soft-limit, one can contact AWS to increase this limit.
- AWS Lambda function supports versioning, each of the different lambda function version as different ARN (amazon resource name).
- AWS lambda function limits

Resources	Limits
Memory allocation	Minimum of 128 KB and maximum of 3008 MB (with a 64 MB increment). Once the limit is reached the lambda function will be automatically terminated.
Ephemeral Storage (temporary storage)	512 MB
Number of file descriptors	1024
Number of process / threads combine	1024
Maximum execution duration	15 min max time-out (default timeout is 3 seconds)
Number of concurrent executions	1000 (soft-limit)
Package maximum size	50 MB

- AWS lambda function monitoring

AWS Services	Monitoring function
CloudWatch	Tracks the following metrics <ul style="list-style-type: none"> • Number of Request • Latency per Request • Number of Request errored out.
X-Ray	Detect <ul style="list-style-type: none"> • Analysed • Monitor & optimized performance issue. • Collects metrics of upstream and downstream system connected to the AWS lambda function. <p>Above information can be used to generate a detailed service graph.</p> <ul style="list-style-type: none"> • That illustrates service bottlenecks, latency spikes, other issues that impacts lambda functions.
CloudTrail	Following parameters can be collected from the cloudTrail metrics <ul style="list-style-type: none"> • Who invoked Lambda function? • When it as invoked • Request and Response parameters • Request timestamp

- The following are the restriction with the Lambda function
 - Inbound lambda function is blocked by Lambda
 - Outbound only TCP/IP protocol is ONLY allowed, in which PORT 25 (*Mail Transfer Agent MTA port is blocked*).
 - pTrace (debugging) system calls are blocked
- All environment variable define in the lambda console are encrypted, using default KMS key however they are NOT store as Cryptic text. To store sensitive information within Lambda console, one needs to use **ENCRYPTION HELPER** which utilized Amazon Key Management Service to store sensitive information as Cryptic text on the console.
- One will get error message in the CloudWatch logs - If function configuration exceeded more than 4KB or environment variable key uses reverved keys for lambda function or the KMS keys is disable which is used for encrypting environment variable.
- **Lambda Runtime Environment:** Lambda runtime is responsible for running the function code setup, reading the handler function name from the environment variable, reading invocation events from the Lambda runtime API and passing it to the lambda handler for processing it. Once Lambda handler process the event data, return back the Lambda functions response back to Lambda runtime API. One can change the Lambda Runtime at any given time form the list of preconfigure lambda runtime environments available.
- Lambda layers – are the zip archives that consists of libraries, custom runtime environment and other dependencies. Lambda functions can leverage Lambda layer to read additional code/content from the lambda layers – this keep the lambda code small, manageable size, easy to debug. Lambda function can use **up to 5 layers**, to **total size of all the unzip code/content (all layers together) cannot exceed more than 250 MB**. It possible to reuse layers publish by AWS or by other AWS customers. Layers uses resource base policies to grant layer accesses to an AWS Organizations, to specific AWS Account or all AWS accounts.
- Common Error Scenarios for Lambda functions
 - Code related error – code runtime executing error.
 - Timeout / running out of memory
 - Concurrency limit reached.
- Lambda Function exceed the default throttling limit for concurrent execution
 - For Synchronous request – it will send HTTP 429 (Too many request Error).
 - For Asynchronous request – it can handle a burst of traffic for 10-15 min beyond that it started rejecting request, for event those are originated by S3 – it will retry for 24 hours before moving it into (Dead Letter

Queue) DLQ. For Kinesis and Dynamo DB streams it retries till the event expires, before moving it into Dead Letter Queue (DLQ).

- Lambda Function Error Handling (AWS Error handling for lambda functions depends on Synchronous execution or asynchronous execution.)
 - For S3 event source – it will automatically retry 3 (three) times
 - Kinesis Stream / Dynamo Stream – it will retry till the data does not expires – default expire of data within Kinesis / Dynamo Stream is 24 Hours. Which can be up to 7 days. NOTE: until the failed batch of record is processed or expired new record will NOT BE PROCESSED from that shard.
 - [FOR NON-blocking processing of streams by lambda functions] : Configure a SQS to pull a batch of records into the queue and Let lambda process the records from the SQS queue, so that in an event of an error – the lambda function returns the messages back to the queue and continue processing of the next message from the queue – once visibility timeout expires the failure message reappears on the queue and the message are reprocessed again.
 - Dead Letter Queue can be configured for asynchronous events once the event retry expires.
- Lambda VPC
 - Lambda function can be configured to connect to the AWS services in a VPC.
 - For enabling VPC support for the Lambda function one need to select one or more subnets and also needs to specify the security group.
 - Lambda function run code securely within the VPC.
 - Lambda function cannot connect to the VPC with dedicate instances tenancy, instead it can be connected with peeing another VPC without dedicated tenancy.
- AWS Serverless Application Model is a specification that prescribe rules for expressing serverless application on AWS, which users CloudFormation syntax and supported natively within CloudFormation to help configuring and deploying serverless applications.
- **AWS Serverless Application Repository**: this is a repository where one can find serverless application build within AWS community by developers, companies, partners.
- To enable **Lambda@Edge**, one need to make function to trigger in response to the CloudFront request (also one need to ensure that it meet **Lambda@Edge** size limits - Currently Lambda Edge only supports node.js (and python) code.
- CloudFront event that triggers Lambda@Edge
 - View Request – when a new request comes from a user / device to the CloudFront URL over HTTPS.
 - View Response – when CloudFront server a request.
 - Origin Request – When CloudFront does not have the requested object on it cache and forwards the request to origin webserver.
- Origin Response – When Origin webserver response back the object in reply to the CloudFront request. (when CloudFront receives reply back from the origin webserver).

Lambda Function Associations

The screenshot shows the 'Lambda Function Associations' interface in the AWS console. It features three main components: a 'CloudFront Event' dropdown menu, a 'Lambda Function ARN' text input field, and an 'Include Body' checkbox. The dropdown menu is currently open, displaying the following options: 'Select Event Type' (with a chevron icon), 'Viewer Request', 'Viewer Response', 'Origin Request', and 'Origin Response'. The 'Lambda Function ARN' field is empty. The 'Include Body' checkbox is also unchecked. A plus icon is visible to the right of the 'Include Body' checkbox.

- Lambda@Edge Use cases
 - A Lambda function can inspect cookies and rewrite URLs so that users see different versions of a site for A/B testing.
 - CloudFront can return different objects to viewers based on the device they're using by checking the User-Agent header, which includes information about the devices. For example, CloudFront can return different images based on the screen size of their device. Similarly, the function could consider the value of the Referer header and cause CloudFront to return the images to bots that have the lowest available resolution.

- Or you could check cookies for other criteria. For example, on a retail website that sells clothing, if you use cookies to indicate which color a user chose for a jacket, a Lambda function can change the request so that CloudFront returns the image of a jacket in the selected color.
 - A Lambda function can generate HTTP responses when CloudFront viewer request or origin request events occur.
 - A function can inspect headers or authorization tokens, and insert a header to control access to your content before CloudFront forwards the request to your origin.
 - A Lambda function can also make network calls to external resources to confirm user credentials, or fetch additional content to customize a response.
- Lambda@Edge Limitation
 - Lambda@Edge cannot access cloud front distribution of another AWS account. Lambd@Edge and CloudFront distribution should be owned by the same AWS Account.
 - Only Lambda@Edge numbered version can be triggered by cloudFront – No aliases version like \$LATEST can be triggered.
 - ONLY lambda function defined in US East (N. Virginia) Region can be triggered by CloudFront.
 - To add triggers, the IAM execution role associated with your Lambda function must be assumable by the service principals `lambda.amazonaws.com` and `edgelambda.amazonaws.com`.
 - Blacklisted headers aren't exposed and can't be added by Lambda@Edge functions. If your Lambda function adds a blacklisted header, the request fails CloudFront validation. CloudFront returns HTTP status code 502 (Bad Gateway) to the viewer.
 - For Read-only headers can be read but not edited.
 - A Lambda function can read, edit, remove, or add any of the CloudFront headers. (`CloudFront-*`) .
 - CloudFront doesn't execute Lambda functions for viewer response events if the origin returns HTTP status code 400 or higher.
 - Lambda@Edge only supports – Nodejs and Python Lambda environments.
 - Functions triggered by origin request and response events as well as functions triggered by viewer request and response events can make network calls to resources on the internet, and to AWS services such as Amazon S3 buckets, DynamoDB tables, or Amazon EC2 instances.
 - Lambda function – CloudWatch Monitoring. By default, AWS monitor multiple AWS Lambda parameter on behalf of the user, like;
 - **Invocation:** Number of time function is invoked in each 5 min period (aggregated for 5 min)
 - **Duration:** The time taken by the function to execute the logic (min/max/ average)
 - **Error Count & success rate %** - number of error encounter and the % of the execution occurred without any error.
 - **Throttling:** Number of times the function failed due to concurrency limit.
 - **Iterator Age:** for streaming workload the oldest item age.
 - **Dead Letter Queue Error:** Number of event AWS Lambda fails to write into DLQ.

AWS SAM (SERVERLESS APPLICATION MODULE)

AWS SAM is an open source framework that can be used for simplifying building serverless applications on AWS. It's a high-level abstraction on AWS CloudFormation template required for building Serverless applications. SAM template is a YML/JSON file with clean syntax to describe functions, API permissions, configuration and events that makes up a serverless applications.

Benefits

- Single stacks – all related resources are deployed using a single template, which makes deployment and management easy.
- Build in Best Practice: Code review, Code deployment using code deploy – enabling X-Ray.

SAM CLI – Command Line Interfaces.

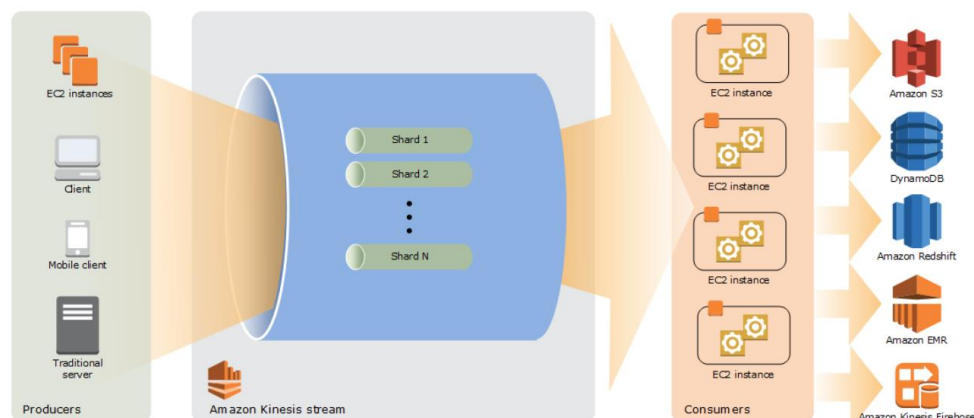
AWS SERVICE – REDSHIFT

- AWS Data warehouse used for Online data analytics processing OLAP NOT for Online Transaction Processing OLTP.
- It's a fully managed petabyte scale data warehouse service.
- AWS Redshift quires are distributed and parallelized across multiple physical resources.
- Redshift encrypts its data at transit, using SSL encryption.
- Redshift encrypts its data at rest, using hardware accelerated AES 256 bits using one of the following means
 - Redshift can manage its encryption
 - Redshift encryption can be managed using KMS service
 - Redshift encryption can be managed using HSM (hardware security module).
- The following are the feature of the Redshift that boost performance
 - **Redshift is a columnar data storage**
 - In Redshift sequential data are stored in columns instead of rows.
 - Columnar data are ideal for data warehouse and analytics
 - This needs less IOPS to read data for analytics operation
 - **Advance compression**
 - Sequential data stored in columnar storage allows better compress then data stored in rows – better compression leads to improve storage.
 - Redshift automatically decides on compression schema that need to be applied.
 - Redshift supports MPP (massive parallel processing) – data and query are distributed access multiple nodes to achieve higher degree of parallel processing.
 - There is no upfront commitment for redshift – one can start small and scale as per the need. Smallest will be of 160GB, single node redshift service, the largest will have up to 128 nodes in single cluster.
 - Typically, in cluster setup – there are multiple nodes. One node will act as a leader node and the remaining nodes will act as a compute node. The leader node will manage connection and receives query whereas the compute node will store data and perform query processing & computation. When there is a single node then the same node act as a leader node and also compute node.
 - Retention & backup: AWS automatically backed up redshift database as per the predefine retention period (default retention period is 24hours, minimum retention period is 0 days maximum retention period 35 days). Snapshot of the backed-up data will be stored as snapshot in S3 bucket. Also, one can chose to take manual backup.
 - When a redshift cluster is deleted – one can chose to create a final snapshot, all other automatic backup will be deleted. However, all the manual back up created will NOT be deleted automatically.
 - **Redshift don't support multi-AZ setup.** One can use backup snapshot to seed a new cluster in a different AZ or region.
 - Redshift monitoring: **Compute utilization, Storage utilization** and **IOPS** can be monitor for free and is available from AWS console and from CloudWatch APIs. Additionally, one can define user define metrics for monitoring custom metrics.
 - Redshift automatically replicate all the data into all the node (except the leader node), on an even of a node failure data can restored back. During the time the node is getting restore redshift cluster will not be available.
 - AWS always recommended to use at least two node cluster, so that data can be restored in case one of the nodes fails.
 - Redshift can asynchronously replicate the snapshot to a S3 bucket in another region for Disaster Recovery.
 - Redshift can be scale at given time, during the time redshift is getting scaled the cluster will NOT be available. As AWS will provision a new cluster with specified size and copy the data from the current cluster to the new cluster.
 - Redshift Pricing:
 - For sum total of all the compute node hours, there is no charges for the leader node hours. Charge as 1 unit per node per hour.
 - For the S3 storage of the manual and automated backup.

- Data transfer charges – any data that is transfer to/from redshift cluster will incur charges as per the data transfer charges. There is no data transfer changes for copy redshift snapshot from redshift cluster to s3 bucket within same region.

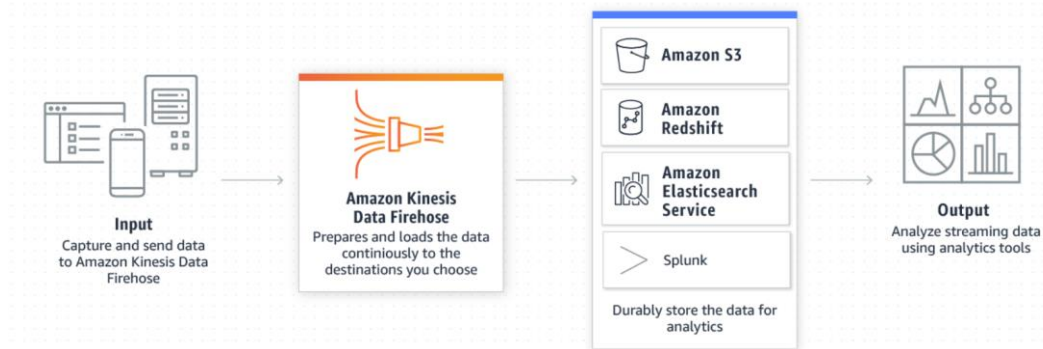
AWS SERVICE – KINESIS

- Kinesis loads large stream of data into AWS
- **Kinesis Stream** – collects the data from the producers and make it available to the consumers.
- **Kinesis stream application** custom application that uses Kinesis client libraries and run on EC2 instances which can reads kinesis stream as data records and process these records to Custom Dashboards, for generating alerts or even use for dynamically change advertisement and pricing policies.
- **Kinesis Data Stream** is a part of Kinesis data platform along with **Kinesis Data Firehose**, **Kinesis Video Stream**, **Kinesis Data Analytics**.
- **Kinesis data stream** can be used for intaking streaming data in real time, other benefits are
 - In case the front-end system fails to read the incoming data still, the data will NOT be lost.
 - Real time metrics and reporting Kinesis data streams can be used to perform simple analytics and processing of the data for custom dashboards – there is no need to batch the record and process using Kinesis stream.
 - Real time data analytics – On Kinesis data stream records analytics can be applied at real time.
 - It can be used for complex data streaming. Output from one stream can be an input to another stream, this helps building a complex data streaming application with multiple upstream and downstream application.



- The main benefit of using Kinesis is to real time aggregation of data followed by loading of the aggregated data into data-warehouse or map-reduce cluster. The consumer application can read data almost immediately without any delay as soon as the data is added to the steam.
- A same kinesis stream can be read by multiple consumer concurrently.
- **Shard** is a sequence of data records –total capacity of the stream, is the sum of capacity of all the shards within that stream. Number of shards can be increase or decrease as per the need. Shard can consume 1Mb/sec rate and produce at 2Mb/sec rates.
- Instead of creating consumer application, **Kinesis Firehose Data Stream** can be used to put the record data directly into Amazon services like – AWS S3, Amazon Redshift, Amazon Elastic Search, Splunk.
- **Retention period:** Time a data record remains accessible once the data is added to the stream – default value is of 24 Hours – can be extended up to maximum of 7 days (168 Hours).
- **Consumer application** – reads the data from the Kinesis stream. There are two types of consumer application
 - Shared fan-out consumers
 - Enhanced fan-out consumers
- Partition is used to group data within shard.
- Server-Side Encryption – Kinesis can encryption sensitive data once producer produce the data to kinesis stream.
- **Kinesis Firehose** – it's an easiest way to load streaming data into Data Lake, Data Store or Analytic tool. It's a part of **Kinesis streaming data platform** It can capture, transform and load data into Amazon S3, Amazon Redshift, Amazon Elastic Search, Splunk enabling real time analytics with the exiting business analytics tool. It

can also batch, compress, encrypt data before loading.



- **Kinesis firehose delivery stream** – Kinesis firehose can be created by creating underline kinesis firehose delivery stream.
- Kinesis firehose to load data into Amazon redshift, it needs to load data first into S3 bucket, then issue a COPY command to load the data from the S3 bucket to Redshift.
- Record is the data that is produced by producers into kinesis firehose delivery stream.
- Kinesis retain a data to max up to 24 hours, if the consumer service is not available the data will be lost.
- Kinesis firehose uses AWS Lambda function for transformation.
- Kinesis Analytics – it's the easiest way to run real time analytics on the streaming data using standard JAVA code or SQL.

Kinesis Video Stream	Kinesis Data Stream	Kinesis Fire Hose	Kinesis Data Analytics
Its use to process/load large amount of streaming video /audio / other data formats into AWS.	Its use to load data from different producer into through consumer application.	Its use to load stream data into AWS service, while performing transformation, compression the data on its way.	Its use to run real time analytics on streaming data using standard SQL and JAVA code.
			It can be integrated with the kinesis data stream and Kinesis Firehose for ingesting streaming data.

AWS TAGS

- AWS tags are managed by the users – it a NAME / VALUE pair (value optional).
 - AWS tags can be used for
 - Cost Allocations
 - Access Control
 - Resource Groups : resources with similar tags group together and can be seen in a single screen.
- <http://jayendrapatil.com/aws-resource-tags/>

AWS SERVICE – EMR (ELASTIC MAP REDUCE)

This topic mostly features in professional exam NOT in associate level exam.

- AWS ERM host web scale Hadoop Framework running on EC2 instance and S3 instances.
- It helps users to spend more time on the analysis then having to do time consuming set for Hadoop components.
- ERM launch all node in a same availability zone, for leveraging benefits due to co-location of the resources.
- ERM charges are on hourly basis, as soon as the ERM cluster is started it starts billing.

AWS SERVICE – SQS

- AWS fully managed queuing service. This is use to build de-couple application.
- Different types of SQS queue
- Message can be of 1KB to 256KB in size
- There is no limit on the number of queues that can be created within an AWS account/region.
- Message can be send/received/deleted in batches – max 10 message per batch max up to 256 KB size.

Standard Queue	FIFO Queue
Available on every region	Only on selected region.
Unlimited Throughput	Limited Throughput (300 message per second).
At least deliver once (There can be duplicate messages send to the end user application)	Exactly once processing (there is guarantee no duplication of message, each message will be processed ONLY once)
Best Effort Ordering (The message can be received in different order then what was send to)	First-In-First Out ordering

- SQS is a polling base messaging

Short Polling	Long Polling
It does not wait for message to be appear	It waits for the max <i>ReceiveWaitTime</i> , before responding back if there is no message in the queue for processing. If there are message for processing then it will respond immediately.
It queries only a small subset of the available servers for message based on the weighted random distribution	It queries all the available servers.
ReceiveWaitTime set to 0 (zero)	ReceiveWaitTime <= 20 seconds

- **Retention period**

- 1 min – 14 days (default is 4 days)

- Once message reached define retention period then the message will be automatically deleting from the queue.

- **Visibility time-out**

- Maximum visibility time-out that can be set to a message is 12 hours.

- Once the message is read – visibility time out is set. Till its expired, the message will not be visible to another consumer.
- Once the visibility time-out is set
 - Consumer successfully process the message the message and send an acknowledgment which deletes the message from the queue so that another consumer does not process it.
 - Consumer seek for extension for visibility time-out, as it taking longer time for processing.
 - Consumer fails to process the message within the visibility time-out duration, the message will be re-appearing in the queue for other consumers to process.

- **Message delay** – A delay of maximum of 15 min can be added, which will delay the messages from publishing into the queue.

- **SQS message reliability** – store message within multiple availability zone within a same region.

- **SQS message security** – IAM policy can be used to control access to the messages within the queue. It supports TLS/SSL connection, SQS is also PCI DSS compliant.

- **SQS supports Server-Side Encryption** – the message once receives in the SQS queue encrypts the messages using KMS manage keys. Once it receives a dequeue message request from an authorized consumer it decrypts the message before sending it back to the consumer. SQS – SSE is available to both standard queue as well as for FIPO queues. SQS– SSE is ONLY available in selected region NOT all regions.
- **In-Flight message:** That are current getting processed by the consumers. There is a **maximum of 12,000 in-flight message for standard SQS queue & 20,000 in-flight message for FIFO queues.**

○ **SQS Message can't be shared across region.**

- **SQS-Monitoring** – basic monitoring is FREE, detailed monitoring is not available.
- **SQS-Logging** – CloudTrail logs all the API calls made to the SQS from AWS API or from the AWS console.

AWS SERVICE – DYNAMODB

- DynamoDB is fully managed AWS No-SQL database, which supports documents and key-value pair.
- Single digit millisecond latency.
- Doesn't supports complex query and join – Doesn't supports complex transactions.
- It's a region-specific service.
- Dynamo DB synchronously replicate the data into three facilities within a same region.
- Dynamo DB uses SSD drives that replicate the data using three-way replication.
- Dynamo DB uses two types of primary key
 - **Partition Key (Hash key):** Simple Primary Key – hash value of the primary decides where the data will be stored.
 - **Partition Key and Sort Key:** Simple Primary Key – hash value of the primary decides where the data will be stored. All value within a same partition key are stored in order of the Sort key.
 - **Secondary index** – adding flexibility to the queries without impacting performance.
- Dynamo DB supports in-place atomic update.
- **Global Table** – Different DynamoDB across different region tied up together, any changes made to one table will replicate to other tables, thereby increasing the durability of the data.
- Based on the required read-write capacity, Dynamodb will scale-out/scale-in.
- Dynamodb supports two types of read consistency
 - **Eventual Consistency (default):** When there is a read after a write the latest data may not be reflected immediately.
 - **Strong Consistency:** Any write operation with success response code, will be fetched when there is read operation after write operation.
- Maximum size of an item = 400KB
- Maximum number of nested attributes = 32 levels.
- **Read-Capacity** = One strongly consistent read per second OR two eventual consistent read per seconds of Item size not more than 4KB in size.
- **Write-Capacity** = Write of 1 KB size of item.
- **If the read/write exceeded then that of the provision capacity, request above the provision capacity will be throttled and the requester will get HTTP 400 (Bad Request) error.**
- **Partition :** <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.Partitions.html>
- Pricing
 - For the provision throughput (regardless of its been used or not)
 - Index data storage
 - Data transfer charges for reading/writing it from outside the region, within region FREE.
- Backup
 - Point-in-Time backup: Will restore the table, but does not create a new a table
 - **DynamoDB maintains continuous point-in-time backups of your table for the last 35 days.**
 - On-Demand backup: Will retain the dynamo db table settings and the data within the dynamo db table.
- Scalability
 - Push Button scalability
 - Any changes made to the read-capacity-units or write-capacity-units will be provision without any downtime
 - **ONLY 4 scale down request can be made for a single calendar day.**
 - **Maximum of 10,000 read-capacity-unit/write-capacity-unit can be set within a DynamoDB table. THIS IS A SOFT LIMIT.**
 - Maximum of 256 table can be created per account per region.

- Security
 - Using IAM role with Fine Grain Access Control (FGAC), one can control which item or attribute of a particular table can be access by whom.
 - Dynamo DB data at transit is encrypted using TSL
 - Dynamo DB data at rest including secondary indexes can be encrypted using SSE leveraging the keys from KMS using AES-256
- Best Practices
 - For storing sequential data into DynamoDB it required action specific to date/time.
 - For storing large item size in DynamoDB, it's advisable to store the object in S3 bucket and maintain an index for that object.

ADVANCE TOPIC

- ✓ **Dynamo DB Secondary Index** from Dynamo DB table the data are retrieved based on the primary attributes, also known as primary key index. Sometimes, the data are need to be retrieved not alone using a primary attribute, thus one needs to create a secondary index. They help in improving the performance and cost by reducing the read capacity. There are two types of secondary indexes –
 - **Local Secondary Index (LSI):** Are the secondary key which shares the same table partition key, and they can be created while creating the table (cannot be added later).
 - **Global Secondary Index:** Global Secondary Index (GSI): Are the secondary key which have different partition key then the parent table, they can be added at any time and have their own provisioned throughput. They are faster, flexible and comes with additional cost AND supports ONLY eventual consistency. Maximum number of Global Secondary Index can be created per table is 20.
- ✓ **Dynamo DB Streams** Dynamo DB Stream, is the transaction logs of the No-SQL table. It records exactly once the changes made to the data in the table. Every time an item added, changed or removed; a stream event is triggered capturing the change into the Dynamo DB change. AWS maintains, separate endpoints for DynamoDB and the DynamoDB streams. There are various use case of the Dynamo DB streams, like moving the No-SQL data from the relational database to execute complex join operation which otherwise couldn't have been possible with Dynamo DB tables. **When Dynamo DB Streams is enabled it hold changes for a single table (NOT entire database).**
- ✓ **Dynamo DB Streams view:** There are four Dynamo DB streams view available
 - **KEY_ONLY** – on the hash key of the change image.
 - **NEW_IMAGE** – the new image of the change item.
 - **OLD_IMAGE** – the OLD image of the change item.
 - **NEW_AND_OLD_IMAGES** – the new and the old image of the change item.
- ✓ **Dynamo DB Triggers:** A lambda function can be integrated to a DynamoDB, which gets automatically invoked when there is any item in the DynamoDB Streams.
- ✓ **Dynamo DB Accelerator (DAX):** Dynamo Accelerator is the fully managed in memory cache for Dynamo DB. It can scale up to millions of requests, with millisecond to microsecond performance improvement. There is no need for changing the application logic, as DAX is compatible with the existing Dynamo DB call.
- ✓ **VPC Endpoints:** VPC endpoint allow access to the dynamo DB from the VPC, the endpoint needs to be created within the same region where the Dynamo DB table.
- ✓ **Dynamo Scan Vs Query Vs Parallel Scan**
 - **Scan** is an operation that scan one or more item or items attributes by accessing all the items of a table or secondary index. Total number of scan items has a maximum size of 1 MB. Scan is a sequential operation, to improve scan performance parallel scan needs to be implemented. Scan results by default are eventual consistency, however one can set ConsistentRead parameter = true to get consistent reading. Avoid scanning large tables or indexes with filter which removes many results.
 - **Parallel Scan**, in order to improve the performance of the scan parallel scan operation can be used instead of Scan operation where, multiple workers run scan operation parallelly. Parallel scans are always returning eventual consistent results and consume high read capacity.

- **Query** operation finds items based on primary key OR using composite key for any table or secondary index. One can set `ConsistentRead` parameter = true for consistent reading, by default query operation returns eventual consistency.
- ✓ **Global Table within Dynamo DB:** AWS managed solution for deploying multi region multi-owner table, which get automatically replicated across all the regions, user need not to maintain application replication logic for replicating the data across regions. To enable global table, one need to follow the following steps
 - Create the desired table (make sure the table is empty)
 - Add Dynamo DB streams to the table
 - Chose the view type
 - Chose Add Region where you want to deploy the table

AWS SERVICE – ECS

- ASW ECS is a fully managed container management service from Amazon, which can be used for Run/Stop/Manage docker container on cluster.
- AWS Fargate is a fully manged serverless architecture for manging docker container clusters.
- AWS ECS can be used to create a consistent deployment and build experience, manage and scale for batch workload or ETL (Extract Load Transform) workloads and build a sophisticated application architecture on microservice model.
- ECS is a Regional Service.
- After ECS cluster is up and running – one needs to define the Task Definition and Service that specifies which docker container services to run across the ECS cluster.
- Docker file is use to create a container image, the container image can be stored in **ECS Container Registry (ECR)** or on the **docker Hub** which can be refed by a service.
- ECS launch Type
 - **Fargate Launch Type:** This help in running containerized application without any need for provisioning and managing backend infrastructure. Just register the Task Definition and Fargate will launch the container. (Fargate Launch Type ONLY supports container images that are store in Amazon ECR or stored publicly on the docker hub. It DOESN'T support images that are stored on private repositories.
 - **EC2 Launch Type:** This help in running containerized application on ECS cluster that user can manage. (Support images from Amazon ECR, from docker hub and also from private repositories).
- **ECS Container Instance:** Amazon EC2 instance with ECS agent installed on it define within an ECS cluster is called as ECS Container Instance.
- **ECS Task Definition:** Task-definition is a JSON base file where one can define up to 10 container definition that form an application. Task definition consist of
 - Which container image to be used.
 - The location of the container image repository
 - The ports that needs to be open for the application
 - The data volume to be used for the containers defined.
 - Other specific parameters available for the task definition depending upon the Launch type.
- **ECS Task:** is an instantiation of the ECS Task Definition. Based on the defined container definition those many containers will be instantiated.
- IAM can be used to manage container instances security by Container Instance Role as wells as IAM Role for the Tasks. Any role define at the Container Instance level, will be inherited by all the tasks. To limit the access, its advice to have minimum access roles define at the Container Instance Role level and define fine grained role at the task level base on the need of the task.
- Benefit of using IAM Role for tasks
 - **Credential Isolation:** A container can ONLY access credentials from the IAM role that it defined in the task definition to which it belongs. It cannot access credentials that are intend to other containers OR tasks.
 - **Authorization:** Unauthorized credentials cannot access IAM credentials that are defined for other tasks.

- **Auditability:** Access and event logs are available for auditing through cloud Trails: Each task has task context taskArn that is attached to the session, CloudTrail logs each of the action task performs and the roles its accessing.
 - **Load balancing ECS tasks using – Classic load balancer:** this can be achieved by having each of the task to run on different port and then configuring CLB to load balance between its listeners. AWS does not recommend to use multiple services with same Classic Loadbalancer as task from one service becomes un responsive – the entire EC2 container instances will get deregister, even if the other service task are still healthy. To overcome this problem one needs to group similar tasks into logical groups call target group and register the target group instead. *[this also avoid high cost, as one can use a single EC2 container instance to host multiple services.]*
 - **Service Load Balancing:** Service Load Balancer can be defined within ECS to distribute load the incoming traffic evenly among the tasks. ALB provide multiple feature to distribute load among ECS tasks.
 - Path base routing
 - Dynamic Host Port mapping
 - Automatically register/de-register new tasks with the ALB
- Dynamic Host Port Mapping: **[need to understand this - better]**
- **Load balancing ECS services using Application Load Balancer (ALB):** ECS service helps maintain a desired count of task at any given time. ECS service can also be configure to use ELB (Classic, Application or Network Loadbalancer) one ELB at a time. Dynamic host port mapping feature make ALB more attractive to use with tasks defined with ECS services.
 - **ESC Service:** is where one can define Application Load Balancer, Classic Load Balancer, Network Load Balancer.

AWS SERVICE – AWS ACTIVE DIRECTORY

- AWS Active directory service provide connectivity with Microsoft Active Directory or LDAP directory services, to use the existing user/group/devices information to access AWS resources.
- **AWS Microsoft Active Directory services** is an AWS managed Microsoft AD services, hosted on AWS side. This is a best choice when there are more than 5000 users and/or need trust relationship between the on-premises & AWS hosted Active Directory. Supports manual and automatic backup (snapshots).

It's a full-fledged Microsoft Active Directory AWS managed service which can be run in replication mode or in trust sharing mode.

To established SSO between the on-premises Active directory and the AWS hosted Microsoft Active Directory, VPN connection is needed.

This can also be used in federated mode, where user can access Office365 on the Azure cloud using AWS hosted Microsoft Active Directory credentials.

Using AWS hosted Microsoft Active Directory, one can login into AWS console without any need for setting SSO between Microsoft Active Directory & AWS IAM service.

Its hosted on two separate AZ within a same region, thus supports HA. One can also add domain controller to scale up the performance of the Microsoft Active Directory AWS managed service.

AWS Microsoft Active Directory services comes in two editions:

- **AWS Microsoft Active Directory Service – Standard edition** This is suite for up to 5000 users, and can manage up to 30,000 objects.
- **AWS Microsoft Active Directory services – Enterprise edition** This is preferred when there are more than 50,000 object that needs to be managed and more than 5000 users.
- **AD Connector:** This connect on-premises Active Directory to AWS. This is best suited when, one likes to connect the existing on-premises active directory to process access to AWS resources. This is a proxy or a gateway for the AD services hosted on premises. It needs a VPN connection or AWS Direct Connect to connect on premises Microsoft AD services for authenticating login users.

There are two types of connector available:
SMALL: which can connect up to 500 users.
LARGE: which can connect up to 5000 users.

IT DOESNOT SUPPORTS SNAPSHOT/BACKUP as there is no data stored within AD connector.

- **Simple AD:** This is the in-expensive solution, a lightweight Active Directory features hosted on AWS. This is best suited when there is NO need for extensive Active Directory features and the user base is less than 5000 users. Supports manual and automatic backup (snapshots).
 This is powered by SMABA 4 active directory compatible server. This can be used for providing access to AWS resources.

This comes with two sizes,

SMALL: which can support up to 500 users and 2000 objects.

LARGE: which can support up to 5000 users and 20,00 objects.

Simple AD does not support following Active directory features like

- DNS Dynamic update
- Schema extension
- Multi factor authentication
- Communication over LDAP
- Powershell AD cmdlets
- FSMO role transfer
- IS NOT compatible with RDS SQL server.
- Trust relationship with OTHER domain
- Active directory Administrative centre.
- Poweshell support
- Active Directory recycle bin
- Group manged service accounts
- Schema extension for POSIX and Microsoft applications.

AWS Microsoft Active Directory Service	AD Connect	Simple AD
Fully functional managed Microsoft Active directory	Used for connecting on-premises Active Directory with AWS	This is a small SAMBA4 base active directory mostly used for manging small user group
>=5000 user <= 5000 users – standard edition. >= 5000 users – enterprise edition.	500-5000 SMALL – 500 users LARGE – 500 - 5000 users	500-5000 SMALL – 500 users LARGE – 500 - 5000 users

AWS SERVICE – CLOUDFORMATION

- AWS CloudFormation is an AWS service for infrastructure as code.
- AWS cloud formation template can be written in JSON or YAML and can be saved as .yaml, .json, .template or .txt
- The main benefits of cloudFormation are
 - Simplify the Infrastructure management
 - Ease of tracking changes to the infrastructure
 - Quick and flawless replication of infrastructure
- There are three major components of CloudFormation
 - Template
 - Stack
 - Change Set

- CloudFormation let one to update a template – once the update template is executed, AWS would generate the change set – once validated the ONLY changes will be implemented.
- If AWS CloudFormation fails to successfully create a resource within a stack, its rollbacks all other changes made successful.
- If a resource is fails to deleted within a stack, no other services are deleted.
- Cloud Formation Designer is a graphical console for building cloud formation template.

AWS SERVICE – CLOUDWATCH

- AWS cloud watch stores cloudwatch logs indefinitely, & however store the Alarm history for 14 days.
- If there is NO data-points associate with the time period – then either Zero OR NULL can be published to the CloudWatch. It's always advisable to publish 0 then publishing NULL. The following are the benefits of publishing 0 (Zero) over NULL
 - If there is NO data points received for specific period adequate alerts can be raised.
 - It's easier to have SUM, MIMIMUM and AVERAGE statistics to be generated with Zero then with NULL.
- **Metrics**
 - Metrics are uniquely identified by name, namespace and optionally by unit of measure.
 - Metrics represents time-order set of data points publish to the CloudWatch, each of the data points have a timestamp and (optionally) unit of measure.
 - AWS Cloud Watch metrics exists in the region where they are created.
 - AWS Cloud Watch does not aggregate metrics across region.
 - AWS Cloud Watch metrics data cannot be deleted, they automatically expire (deleted) by AWS.
 - For 1 min data points metrics data are retained for 14 days
 - For 5 min data points metrics data are retained for 63 days (9weeks)
 - For 1 hr (60 min) data points metrics data are retained for 455 days (15 Months)
- **Namespace**
 - It's the container for the metrics
 - Metrics with different namespace are isolated from one another, so that metrics data from different services does NOT get aggregated into same statistics
 - There is NO DEFAULT namespace, each data element put into AWS cloud watch must have a namespace associated to it.
- **Timestamp**
 - Each data points must have an associated timestamp, if there is NO timestamp is provided with the data point, CloudWatch add the timestamp based on the time it receives request.
 - **The CloudWatch Timestamp can be 2 weeks in past and up to 2 hours in future.**
- **Dimensions**
 - Every metrics has a characteristic, and the dimension are the category of those characteristics.
 - Dimensions can be used to filter the result set that CloudWatch returns.
 - Up to 10 dimensions can be attached to single metrics.
- **Unit**
 - Unit of measure
- **Period**
 - Length of time.
 - Although the periods are express in seconds, the minimum granularity of a period is 1 minute.
- **Aggregation**
 - CloudWatch aggregates statistics based on the period mention in the getMetricsStatus.
- **Statistics**
 - Statistics are metrics data aggregation over a period of time, which are aggregated over name, namespace, timestamp, period, dimensions, and the unit of measure.
 - CloudWatch DOES NOT aggregates metrics data across region.
- **Alarms**
 - Based on define parameter, alarms automatically act on behalf of the user.

- Alarm watch a single metrics and perform one or more actions based on the value of the metrics for a given threshold over the number of periods.
- Alarm is activated ONLY when a metrics value change and maintained for a specific number of periods.
- Alarms CAN
 - Raise SNS notification
 - Act on autoscaling policy
 - Stop or Terminate EC2 instances
- Alarms States
 - ON
 - ALARM
 - IN SUFFICIENT DATA
- **Alarms exist ONLY on the region where alarm is created.**
- **Alarm action should also be in the same region where Alarm is created.**
- For testing an ALARM one can use `setAlarmState` API
- Alarm can be enabled or disable by `disableAlarmAction` and `enableAlarmAction` API.
- **Custom Metrics**
 - Cloud Watch allows putting custom metrics using CloudWatch CLI (`put-metric-data`). If `put-metric-data` method is called with a new metric name then the CloudWatch aggregates the metric data under a new metric name else it adds to the existing metric name.
`put-metric-data` CLI command can add a single metric data at a time. Each data point added using `put-metric-data` command is associated with a specific timestamp.
 - It can take up to 2 min for a statistic can be retrieved using `get-metrics-statics` command on a new metric added through `put-metric-data` command AND up to 15 min for the metrics to be available to be listed using `list-metrics` command.
 - Custom Metrics data points can be published as
 - **Single data point**
 - In case of the single data point, the metric data can be published as granular as 1/1000 of a second, but the minimum aggregated granularity will be 1 minute.
 - CloudWatch records – average value received per min, total sum of the values, maximum value and minimum value for 1-minute period.
 - CloudWatch use 1-min boundary to aggregate data points.
 - **Aggregated data point (or Statistic Set)**
 - Data-points can be aggregated and then publish to CloudWatch
 - This minimized the number of calls to me made to the CloudWatch to publish the data-points.
 - Statistics includes – AVERAGE, SUM, MINIMUM, MAXIMUM and the SAMPLE COUNT.
- CloudWatch Monitoring for EC2 Instances
 - By default, basic monitoring is enabled for all EC2 instances which sends EC2 metric data for every 5 min without any additional cost.
 - One can also enable detailed monitoring for any EC2 instances which send EC2 metrics data for every 1 min with will incur additional cost.
 - CloudWatch provide most of the EC2 metrics out of the box, however any metrics related to memory utilization and disk space utilization are NOT available.
- RDS database related information are NOT available on the CloudWatch (they can be view directly on the RDS console). CloudWatch collects and process raw data from the RDS console into readable, near real time metrics. The data are store for 2 weeks (14 days) for one to view the historical data.
- Here are the list of AWS services and their respective metrics capture by CloudWatch :
<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/aws-services-cloudwatch-metrics.html>

AWS SERVICE – BUDGET

- AWS Budget Dashboard helps in creating, tracking, inspecting budget allocation for different AWS resources.

- Budget can be created on monthly, yearly, quarterly basis – with the ability to adjust the start-date and the end-date and refine cost allocation based on multiple dimension such as AWS services, linked accounts, tags and others.
- In AWS Budget one can set reservation utilization and coverage targets and received alerts when the utilization drops below the set threshold. Reservation Alerts supports – Amazon EC2 instance, Amazon RDS, Amazon Redshifts, Amazon Elastic search and Amazon Elastic Cache.

AWS SERVICE – COST EXPLORER

- AWS Cost Explorer helps in visualizing and managing the cost based on different AWS services, it also helps in viewing cost related data for up to 13 months and also helps in forecasting cost for the next 3 months along with recommendation to reserved instance to purchases. It also helps in identifying the areas that needs further inquiry and see the trends to understand the cost.

AWS SERVICE – OPSWORKS

- AWS OpsWorks is a chef base configuration management tool
- It consists of Cook-Book each of the cook book consist of one or more recipes. A recipe is a set of one or more instruction written in ruby language syntax – which define resource to use and the order in which the resource to be applied.
- **AWS OpsWorks stack** is a collection of resources (AWS resources) group together to cater a need for a specific application.
- **AWS OpsWorks layer** is a set of EC2 instances that servers a particular purpose. Like for hosting database server or for hosting web application etc. Layer give complete control on the package that needs to be installed , how they need to be configure and how application are deployed.
- AWS CookBooks – are store in amazon s3 or in git repositories.
- AWS stack certain properties can be updated using recipes and some of the properties like AWS region cannot be changed. One need to delete the stack and re-create it one a different region.
- AWS OpsWorks lifecycle
 - **Setup** – once a new instance is successfully booted
 - **Configure** – once the instance enters into ONLINE state (ready for deployment)
 - **Deploy** – Deploy an app
 - **UnDeploy** – Undeploy an app
 - **Shutdown** – When an instance is deleted
- Once a lifecycle state is reach – AWS Ops will automatically run the recipes for that state.
- Ops work can control on-premises computes and stacks.
- Ops works Instance: Represents a single compute resources, its comprises of AWS EC2 instance as well as on-premises servers.
- Ops works deploy an agent on the instance through which it controls & manages the instance.
- Ops Works instance type
 - 24/7 instances – these are the instance (compute instances) those are started manually and remain started till its shutdown manually.
 - Time-Base instances – these are the instance that are started by OpsWorks based on a specific schedule.
 - Load Base instance – these are the instances that are started by OpsWorks based on the load
- OpsWork auto healing feature is triggered when OpsWorks doesn't find response/able to communicate with the OpsWorks agent installed on the instance it restarts the instance. In case a single instance is used within multiple layer , and in one of the layer auto healing is turn off then auto healing will not occur.

AWS SERVICE – STORAGE GATEWAY, AWS EXPORT/IMPORT, VM EXPORT/IMPORT

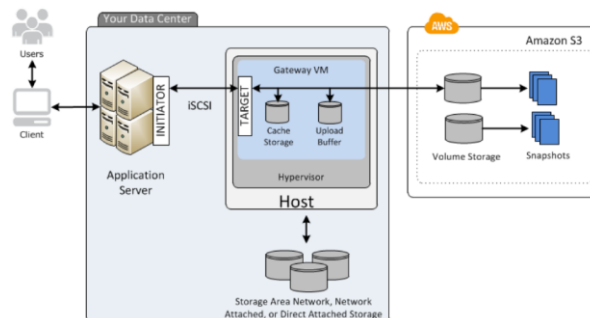
○ Storage gateway

- **File Gateway:** Automatically backed up the files from on-premises application on to the S3 bucket while caching the frequently access files. It uses **NFS (network-file-system) interface** as wells as **SMB (Server Message Block)** for file transfer. **THIS IS MOSTLY USE AS STORAGE OPTION.**



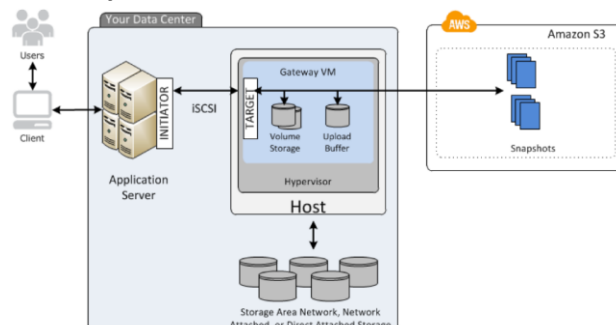
- **Volume Gateway:** Volume gateway transfer block storage for on-premises application using **iSCSI (internet Small Computer System Interface) protocol** to the S3 bucket on the cloud. Data are read as blocks and will be move to the s3 bucket over SSL channel. There is no direct way to access data from the s3 bucket, one need to create an EC2-snapshot from the data then use it to create a new EC2 volume in-order to access the data. **THIS IS MOSTLY USE AS BACK-UP OPTION.**

▪ Gateway Cache Volume

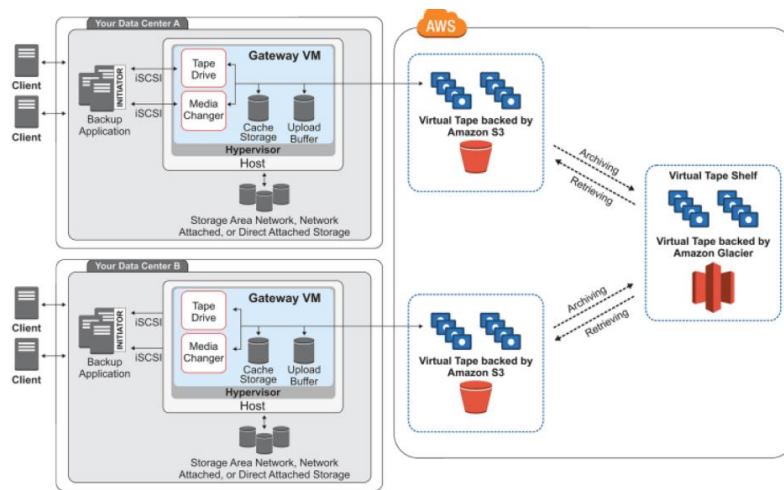


- Here the on-premises data are transferred and stored in S3 buckets, while frequently use data are cached for accessing it with less latency.
- Each of the Gateway volume can have up to 32 individual volume, and each volume can range from 1GB to 32GB

▪ Gateway Store Volume



- Gateway store volume maintains the entire data locally as well as store a backup copy on the S3 cloud bucket.
- Each of Gateway can have up to 32 volume, and each volume range from 1GB to 32GB.
- **Tape Gateway:** Provides backup for the application with the iSCIS virtual tape library interface, consisting of virtual media changer, virtual tape library VTL and virtual tape shelf VTS. **THIS IS TAPE BACK-UP STORAGE.**



Storage gateway upload the data to S3 over SSL channel and provide data encryption when store under S3 bucket over ASE-256 encryption.

- AWS import/export – in order to migrate a large amount of data from on-premise application to the cloud. SNOWBALL which is large secure hard-disk which can ship to the client address and then ship back to AWS location for uploading it to the cloud.
- VM import/export - to import virtual machine to/from AWS. ONLY those EC2 instance that were initially created using importing VM can be exported back from the AWS. There is NO additional cost for VM import /export.

AMAZON -MQ

- Amazon MQ is an industry standard API protocol for messaging including – JMS, .net Messaging Service (NMS), AMQP, STOMP, MQTT, open wire and web socket.

AWS-STEP FUNCTIONS

AWS Step function is a serverless orchestration for modern application using a virtual workflow, its build on the concept of state machines and tasks. Where tasks are accomplished by an activity, or by a Lambda function or by-passing specific parameters to an AWS services, while state machine maintain the state, their relationship and input/output of the state.

AWS Step function uses ASL (Amazon State Language) which is JSON like language which can be use to define a STATE MACHINE a collection of States. Statelint tool can be used to validate the Amazon State Machine.

AWS Step Functions	AWS Simple WorkFlow (SWF)
It's a 100% serverless AWS managed services without any needs for the uses to manage/maintain any underline resources. Initially the services were ONLY supporting AWS lambda functions but later its support were extended to – AWS Batch, AWS DynamoDB, ECS, Fargate ECS, SNS, Glue Stagemaker. It can also integrate to any HTTPS endpoints.	AWS workflow application integrated any task that are running on EC2 instances, on-premises servers or even running on a different cloud provider. AWS SWF ONLY maintains the state machine, while users are responsible to maintain the underline task infrastructure.
In case of the Step Functions, the workflow logic is defined within the step function services.	In case of simple workflow services, task execution are define in activity worker , and the flow login is define under decider worker application code. NOTHING is defined within the SWF.
Workflow logic can be defined and represented using virtual workflow GUI within the step functions.	Workflow logic is defined under the application code, NOT within the SWF.

Entire workflow can be defined with CloudFormation template and AWS serverless application model.	ONLY EC2 instance can be defined using CloudFormation template.
Support for long running processes – typically supports a process for 1 year but limited by the lambda functions limits – 15 min max execution.	100,000 workflow per domain (100 domain max). Max workflow execution time – 1 year Max workflow retention – 90 days Max event history 25,000 events

Source : <https://searchaws.techtarget.com/tip/Step-Functions-outshines-SWF-for-most-AWS-workflows>

AWS Step Functions	AWS SQS
Ideal for highly scalable solution and auditable solutions	Idea of solutions which needs – highly scalable, reliable, hosted queues for communicating between different services.
Step function can keep track of the different tasks and the event that has occurred during the flow. Step function console provide an application centric view, where one can view the executed steps, search for a specific execution, drill down on an execution task.	Using SQS one needs to develop custom logic to keep track of the tasks and event that has taken place during the flow. In case of SQS one need to build such functionality, these functionalities are NOT available out of the box.

AWS SERVICE – IAM (IDENTIFY AND ACCESS MANAGEMENT)

- AWS IAM webservice manages access control, authorization and user management for other AWS services.
- **IAM feature:**
 - IAM services helps in maintaining **Share Access** of the AWS account with other users without sharing the username and password of the root account.
 - IAM services helps in providing **Granular Permission**, to maintain a minimum access permission policy
 - IAM services helps in providing **Secure access to AWS resources** for application running on EC2 instances (*role base access*).
 - **Multi Factor Authentication:** IAM Services provides an additional layer of authentication apart from the username & password, where use needs to share authentication code from a multi factor authentication devices (can be a software or a physical device).
 - **Identify federation:** Allow authenticated users from different trusted corporate or internet authenticator to access AWS resources.
 - **PCI-DSS compliant:** AWS IAM service is PCI-DSS compliant service.
 - AWS IAM web services provide **Eventual Consistency**, it doesn't support strong consistency.
 - **Security Token Service (STS):**
 - There is **No Additional** charges for using AWS IAM services, user ONLY needs to provide fees for access the other AWS resources.
- How to access AWS IAM services?
 - IAM services can be access from AWS web console.
 - Through programmatic access for CLI, HTTPs, SDK base APIs– where user need to share access key and secret-access-key, to authenticate themselves with IAM services.
- **Element of IAM**
 - **Principal** is the role/user/application/federated users who is trying to access the request.
 - **Request** - request by the principal to access the resources.
 - **Authentication** – user having access the resources, these are based on the policies – there are two types of policies **Identity Base Policy** and **Resource Base Policies**.

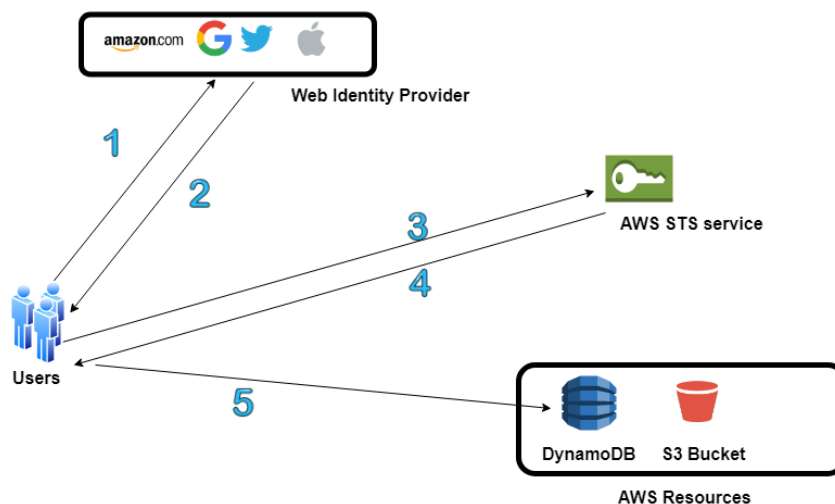
- **Authorization** – User is authorized to access the resources. If there is explicit denies then, it will override all the implicit allows. If there is explicit allows then, it will override all the implicit denies.
- **Action** – What actions are allowed to the user to perform.
- **Resources** – the resource itself (AWS services).
- **Cross-Account-Access:** Identity base policies can be used for accessing resources within same account. For cross account access, account that owns the resources (Trusting account) need to share access with the account which users need access to the resources (Trusted account). Once the access is granted by the Trusting account, Trusted account can then share/delegate the access to its users.
- IAM Identity Federation: Allowing authenticated users log in into AWS service through trusted authenticator like corporate LDAP or internet identity federated services. Where users can access the account using temporary identity (STS = Security Token Services) shared by the AWS service on behalf of the trusted (federated) identity provider.
 - No need to migrate users from the corporate directory user into AWS IAM service.
 - User using internet identity then application can leverage the same identity for providing access.
 - For providing SSO (single sign on) access.
- **IAM identity – ROLE/GROUPS/Users**
 - Users will have one pair of username/password & up to two pairs of access keys and secret access keys to access AWS service programmatically.
 - Group is a collection of the users, which have similar access (up to 300 groups can be created per account and a user can be a member of up to 10 groups).
 - Similar to a user but NOT assigned to any individual user. Any user can have the roles to perform a specific operation.
- IAM users doesn't have any access to the by default, specific policies need to be attached in-order to specify access to specific resources.
- Resource based policies – where principle will be defined within the resources and will NOT be attached to a user/group/role. It can be used to grant access to the following AWS services
 - S3 – Bucket ACL/ Object ACL
 - SQS
 - SNS
 - Lambda Function
 - Ops Works Stacks
 - Glacier Vault
- IAM users are global entity – there are not regional constructs. Each IAM user are uniquely identifiable by arn (amazon resource name). User can list, rotate and manage their own access keys through IAM policies NOT through console menus. Once an access key is lost, one needs to recreate a new access key, there is no provision to retrieve OLD access keys.
- IAM roles can be assumed by any users or application to received access to the resource, they DON'T have user-name and password instead they rely on STS. When a user switch role, it loses the original access, as it takes up the new access. When user goes back to the original role, old accesses are restored back. At a time ONLY one role can be attached.
- Cross-account-policies using resource base policies: When users access a cross account resources using resource, base policies instead of role base policies – the benefit is the user does not need to give away the access/permission of other AWS account while gaining access to the new account resources. Limitation is NOT all resources has resource base policy.
- IAM services roles: IAM Role for AWS service to access another AWS service, this remove the need for storing & maintaining AWS credentials in another AWS service. IAM would assign a temporary credentials to access the AWS service and also rotate the credentials to keep it access available. **ONLY one role can be assigned to an AWS service at a given time, all application running on the AWS service will share the same role.** However, in case of ECS EC2 instances – multiple task running on the EC2 instance can have multiple IAM role.
- EC2 instance profile is where the IAM ROLES will be attached to the EC2 instance. From EC2 instances one can view the role through metadata service
<https://169.254.168.254/latest/meta-data/iam/security-credentials/ec2role>

- IAM Role delegation: There are two policies need to be attached in order to delegate access role from one AWS account (Trusting account) to another (Trusted account).
 - Permission policy (JSON based policy) should be attached that defines the permission to the principle defined in the trust policy.
 - Trust policy will have the Trusted Account principle to which trusting account grant permission. It can't have wildcard character as principal.
- Cross account access.
- IAM logging can be done using Cloud Trails – based on specific AWS service region base sing-in URL can be possible.

Note : if one needs to restrict the access of certain or all users to change/rotate password, then uncheck the option from all users to change/rotate password by login into IAM console and then write a policy to allow user to change the password and assigned it to users who needs the access to change password.

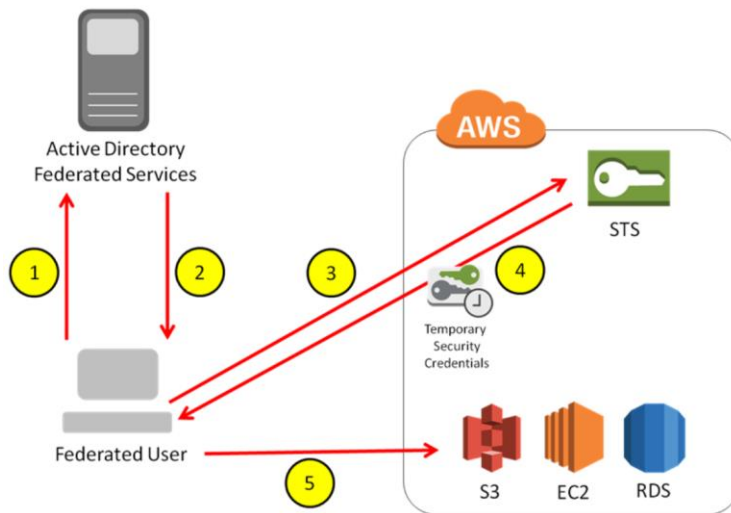
STS (Security Token Service)

- STS is a global construct, however there are certain region where one need to manually enable the STS service, and for the rest of the region the STS service is enable by default.
- For those regions where one need to manually enable the STS service, the call is directed to the region-specific URLs AND for those regions where STS is enabling by default calls are directed to the global endpoint.
- In case of the cross-region STS, it's the region of the calling account (Trustee Account) where one needs to enable for the STS. Once the token is received it can be used globally. There is no need to enable STS for the Trusted account to use STS token.
- *STS can be generated ONLY by one of the SDK or the CLI /Microsoft Power shell scripts. IT can't be generated using AWS web-console.*
- How Web Identity Web Federation works using public IdPS (Identity Provider Service)?



Note: Here AssumeRoleWithWebIdentity API will be called

- How Enterprise Identity Federation works?



Note : Here AssumeRoleWithSAML API will be called

- There are multiple STS APIs to request for the STS tokens, based on this API the life-span of the STS token is decided between 12-36 hours.

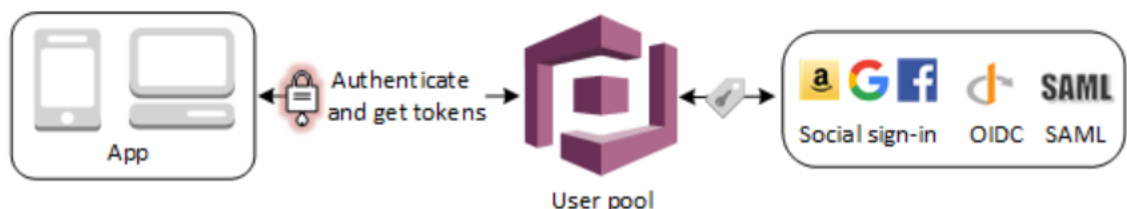
API call for STS	Description
AssumeRole	<p>This returns a temporary security credentials to access AWS resources for which one may not able to access her resources normally. Temporary access consists of Access Key ID, Secret Key, Security Token. It also supports Pass policy, where one can pass a policy based on which AWS will decided whether the user will have access or not.</p> <p>It can be used by any IAM user or any user who have temporary access.</p> <p>This support MFA authentication.</p> <p>Max life span of the token is up-to 1 hr. Minimum of 15 min</p>
AssumeRoleWithSAML	<p>This is used for establishing SAML 2.0 based federation between third party provided and the AWS IAM service.</p> <p>It can be used by any user that can pass an SAML authentication response that indicates authentication from a known identity provider.</p> <p>Max life span of the token is up-to 1 hr. Minimum of 15 min</p>
AssumeRoleWithWebIdentity	<p>Any user that can send web authentication response that indicates authentication from a known identity provider.</p> <p>Max life span of the token is up-to 1 hr. Minimum of 15 min</p>
GetSessionToken	<p>This is used for MFA authentication. This is used by the IAM user or the root user to request token based on MFA authentication token.</p> <p>Life span of the token is from 12Hr up-to 36 hours.</p>
GetFederationToken	<p>This is used by IAM user or the root user.</p> <p>Life span of the token is from 12Hr up-to 36 hours.</p>

- When there is a policy attached to a role, while assuming a role there can be another policy attached to it than that policy will be bringing into effect NOT the policy that is initially attached to the role.

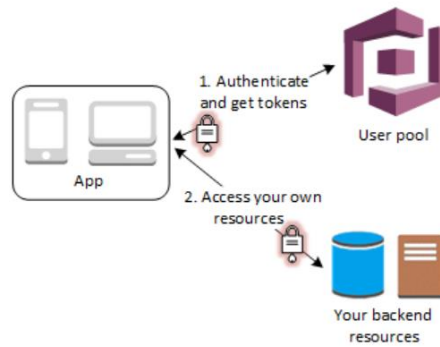
- IAM Certificate Store: Ideal place to store CA signed certificates will be in ACM (AWS Certificate Manager) for those regions where ACM is not available, another alternative is to use IAM Certificate Store to store CA certificates to be use for different AWS services. One can't use IAM console (AWS console) to load a CA signed certificate to IAM Certificate Store, it needs to be done by CLI.
NOTE: AWS CloudFront can be loaded with the CA certificates, which can't be use by other services. It will be tied up to a single CloudFront distribution.

AWS COGNITO

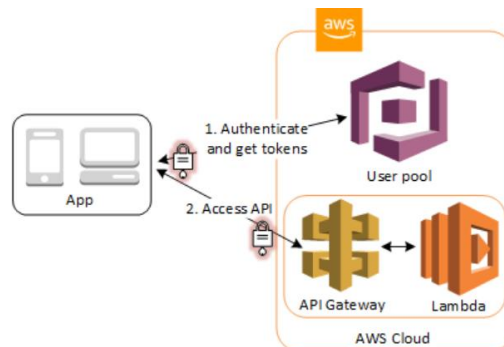
- AWS Cognito can be use for authentication, authorization and user management for web and mobile application users.
- User can sign-in using username password or through Federated 3rd party Identity Provider credentials.
- The two main component of Cognito are, they can be use together or separately.
 - **User-Pool** provide user sign-in and sign-up options. The following feature are provided by the Cognito user pool.
 - Sign-up and Sing-in services.
 - Built-in, customizable web-ui for sign-up users.
 - Sign-in through 3rd party identity provider.
 - User directory management / User profile management
 - Security feature such as MFA, check for compromised credentials, account takeover protection, phone and email verification.
 - Customizable user management workflow through AWS lambda triggers.
 - **Identity-Pool** Can be used to grant access to the other AWS services. With Identity pool, the user can receive temporary AWS credentials for accessing various AWS services. Identity pool can work with anonymous user, or users from the identity providers
 - Amazon Cognito user-pools
 - Social sign-in from web identity providers like Amazon, facebook, google,
 - OpenID Connect (OIDC) federated users.
 - SAML Identity provider
 - Developer authenticated identity.
- To save the identity, Cognito identity-pool need to connect to Cognito user pool.
- The following are the six use-case where Amazon Cognito can be used.
 - **Authenticate using user-pool:** Amazon Cognito can be use to sign in directly or through openID using third party identity provider services. Amazon Cognito take care of the overhead of the connecting to the third-party identity provider and authenticate user and receive authentication token. Once login successfully Amazon Cognito will return user pool token, which can be used to provide access or access other AWS services.



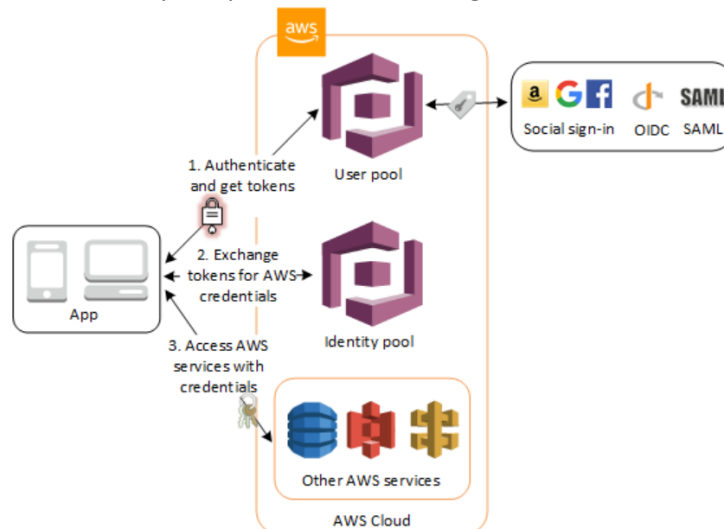
- **Accessing server-side services using user pool:** Once successfully authenticated, using the user pool token one can provide access to the backend AWS services. One can also manage user-pool-groups for managing authentication and different type of users.



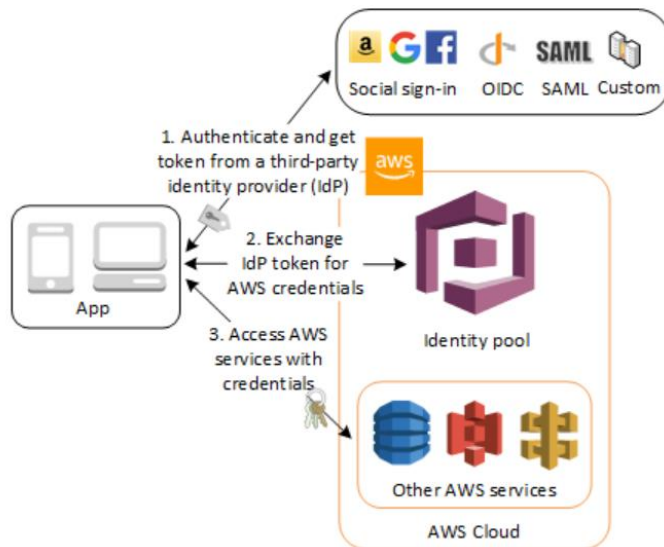
- **Access Resources with API gateway, Lambda with user-pool:** Once user is authenticated, and received Cognito user pool token, it can be forwarded to API Gateway for authentication and to provide access to the Lambda function or custom endpoint. One can also use user-pool groups for managing permission.



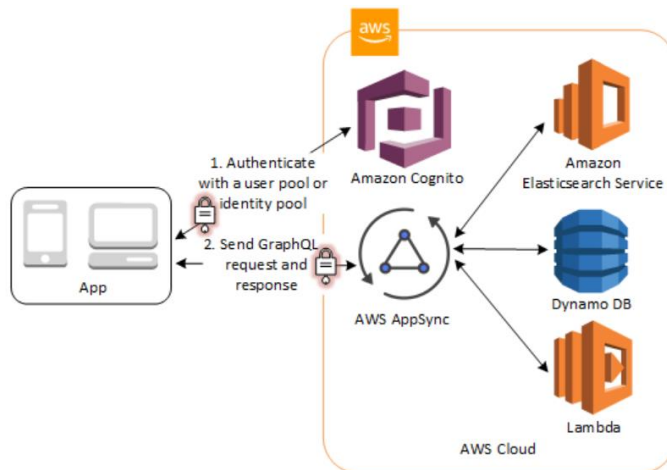
- **Accessing AWS services using user-pool and identity-pool:** User can login into Cognito, on successful authentication user will receive Cognito user-pool token which can be exchanged with identity-pool to receive a temporary access for accessing other AWS services.



- **Using third party identity provider with identity-pool:** One can directly use third party identity provider in conjunction with identity pool for providing temporary access to AWS services.



- **Access AWS App Sync resources with Cognito:** Amazon Cognito Sync, enable syncing of application related user-data across multiple mobile devices without any need for writing the custom code for syncing. Client libraries read and writes data locally regardless of connectivity, once the device is online Cognito App Sync will sync the data and it push-sync is configure it will also push the synced information to the other devices.



AWS ORGANIZATION

- AWS Organization helps in managing multiple accounts. It helps in
 - Centrally manage policies across multiple AWS Accounts.
 - Control access to AWS service.
 - Automate AWS account creation and management.
 - Consolidate billing across multiple account.
- One need to attach AWS account to a specific organization unit.
- **Service Control Policy (SCP):** Allow or Deny a specific AWS services to be used within an organization. SCP override the IAM polices. BY default, SCP will be disable, one needs to enable it from the root account before applying SCP to any organization.
- **FullAWSAccess** is a service control policy that allows users to access services/resources on an attached account.

AWS SERVICE COMPARISON

S3 Select	Athena	Redshift Spectrum
This allows to retrieve subset of data from S3 storage using SQL queries as a result improve data retrieval performance	This helps to run SQL query against S3 stored data.	This helps in running Redshift quires directly on the exabyte of data in S3 with any need for ETL process