AWS NOTES- Part C

Contents

AWS Service – Application Load Balancer 1

AWS Service – Network Load Balancer 3

AWS Service – ELB Comparison 3

AWS Service – Elastic Cache 4

AWS Service – API Gateway 6

AWS Service – AWS Lambda 7

AWS Service – Redshift 9

AWS Service – Kinesis 10

AWS Service – EMR (Elastic Map Reduce) 12

AWS Service – SQS 12

AWS Service – DynamoDB 14

AWS Service – ECS 16

AWS Service – AWS Active Directory 17

AWS Service – CloudFormation 18

AWS Service – Cloud Watch 18

AWS Service – Budget 21

AWS Service – Cost Explorer 21

AWS Service – OpsWorks 21

AWS Service – Storage Gateway, AWS Export/Import, VM Export/Import 22

AWS Service – IAM (Identify and Access Management) 24

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
  - **Application Load Balancer:** The load balancer component that route the incoming traffic to its 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.

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.

AWS Service – ELB Comparison

- Comparison of three Elastic Load Balancers.

Feature	**Application Load Balancer**	**Network Load Balancer**	**Classic Load Balancer**
Protocal	HTTP/HTTPs /HTTP2/Websocket	UDP/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
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 TSL/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	NO	**NO**
Private Link Support	**NO**	YES (TCP,TSL)	

Feature	**Application Load Balancer**	**Network Load Balancer**	**Classic Load Balancer**
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.
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 it 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 severed 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 send back to the request 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 – API gateway supports CORS, it needs to be enable at method level.
  - Provide API gateway dashboard

AWS Service – AWS Lambda

- AWS compute service without any need for provisioning and maintaining servers.
- 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 trigger –
  - 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 service 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
  - AWS Alexa
  - AWS Lex
  - AWS API Gateway

    - 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 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 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	300 seconds.
Number of concurrent executions	1000 (soft-limit)

- AWS lambda function monitoring

AWS Services	**Monitoring function**
CloudWatch	Tracks the following metrics • Number of Request • Latency per Request • Number of Request errored out.
X-Ray	Detect • Analysed • Monitor & optimized performance issue. • Collects metrics of upstream and downstream system connected to the AWS lambda function. Above information can be used to generate a detailed service graph.

	• That illustrates service bottlenecks, latency spikes , other issues that impacts lambda functions.
CloudTrail	Following parameters can be collected from the cloudTrail metrics • Who invoked Lambda function • When it as invoked • Request and Response parameters • Request timestamp

- 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 revered keys for lambda function or the KMS keys is disable which is used for encrypting environment variable.
- Optimizing

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 wearhouse 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 datawarehouse 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 upto 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

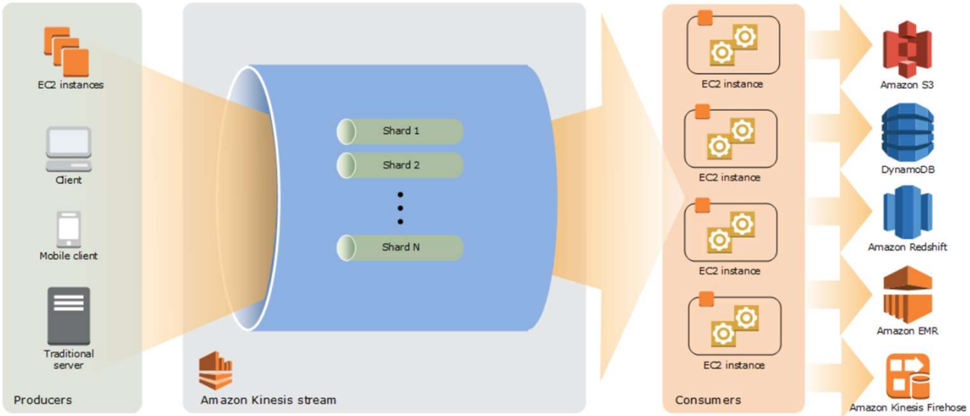
connection and receives query whereas the compute node will store data and perform query processing & computation. When there is a single node then the same node act as a leader node and also compute node.

- 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 node 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

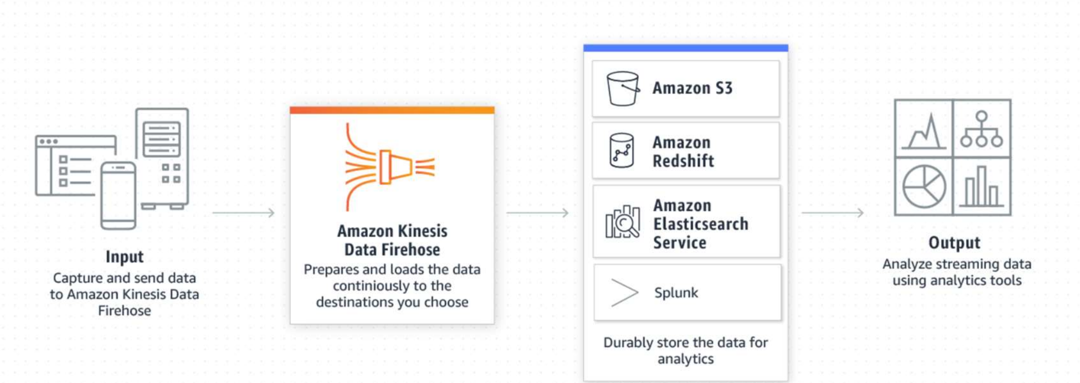
- Kinesis loads large stream of data into AWS
- **Kinesis Stream** – collects the data from the producers and make it available to the consumers.
- **Kinesis stream application** custom application that uses Kinesis client libraries and run on EC2 instances which can reads kinesis stream as data records 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.


Input
Capture and send data to Amazon Kinesis Data Firehose
Amazon Kinesis Data Firehose
Prepares and loads the data continiously to the destinations you choose
Amazon S3
Amazon Redshift
Amazon Elasticsearch Service
Splunk
Durably store the data for analytics
Output
Analyze streaming data using analytics tools

- **Kinesis firehose delivery stream** – Kinesis firehose can be created by creating underline kinesis firehose delivery stream.
- Kinesis firehouse 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 Service – EMR (Elastic Map Reduce)

To be added later

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 ReceivedWaitTime, 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.
ReceivedWaitTime set to 0 (zero)	ReceivedWaitTime <= 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 a acknowledgments 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 TSL/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.
- Pricing
    - For the provision through put (regardless of its been used or not)
    - Index data storage

  - Data transfer charges for reading/writing it from outside the region, within region FREE.
- 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 dyanomoDB 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 Dyanomo DB it required action specific to date/time.
  - For storing large item size in Dyanomo DB, its 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 know 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 and Global Secondary Index.
Local Secondary Index: 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: 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.

✓ **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.

✓ **Triggers**

✓ **DAX**

Dynamo Accelerator is the fully managed in memory cache for Dynamo DB. It can scale up to millions of request, 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.

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 a ECS cluster is called as ECS Container Instance.
  - **ECS Task Definition:** Task-definition is a JOSON 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. Base 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.

- **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 Service – CloudFormation

- AWS CloudFormation is a 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, it 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. Its 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.
  - Its 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 use to filter the result set that CloudWatch returns.
  - Up to 10 dimension 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.

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.

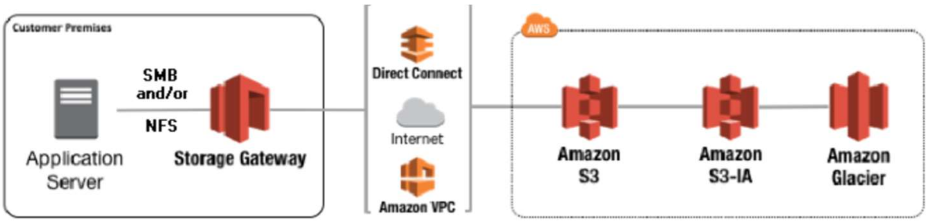
AWS Service – OpsWorks

- 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)

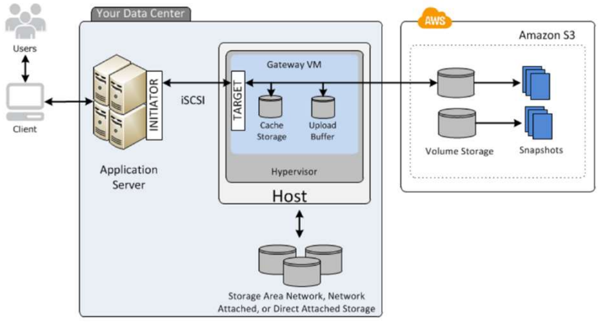
    - **Deploy** – Deploy an app
    - **UnDeploy** – Undeploy an app
    - **Shutdown** – When an instance is deleted
- Once a lifecycle state is reach – AWS Ops will automatically run the recipes for that state.
- Ops work can control on-premises computes and stacks.
- Ops works Instance: Represents a 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-primes 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.**

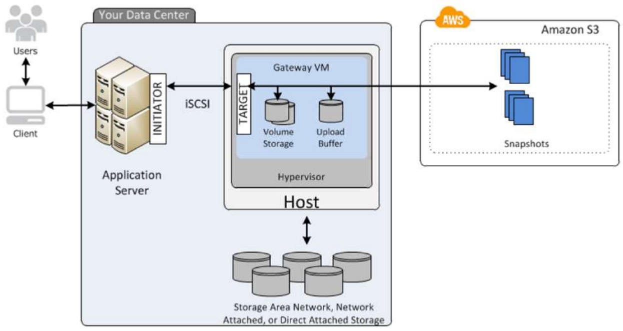


    - Volume Gateway: Volume gateway transfer block storage for on-premises application using ***iSCSI (internet Small Computer System Interface) protocol*** to the S3 bucket on the cloud. Data are read as blocks and will be move to the s3 bucket over SSL channel. There is no direct way to access data from 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**



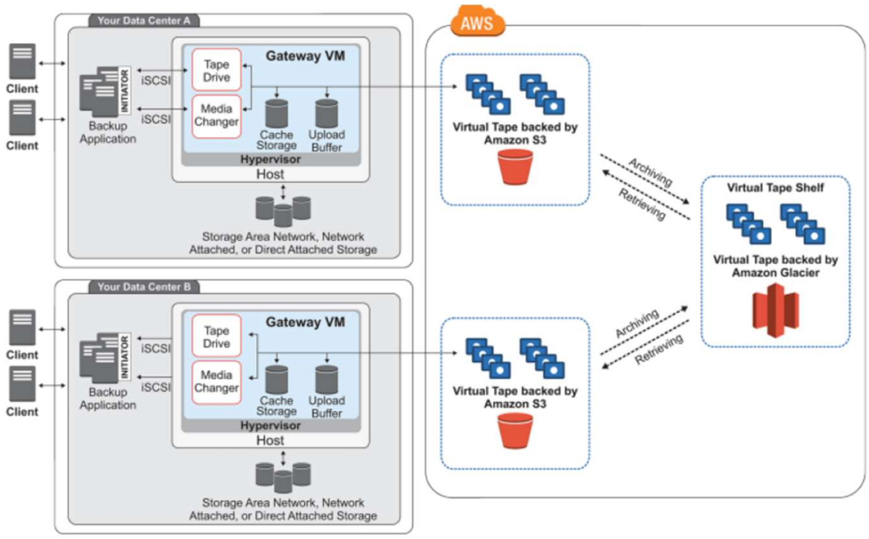
- Here the on-premises data are transferred and stored in S3 buckets, while frequently use data are cached for accessing it with less latency.
- Each of the Gateway volume can have up to 32 individual volume, and each volume can range from 1GB to 32GB

▪ **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.

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 polices 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.

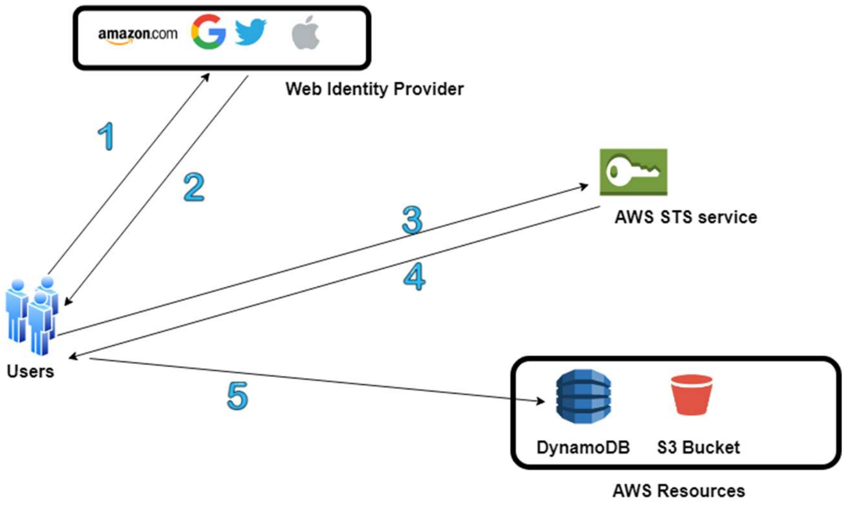
- **IAM Identity Federation: Allowing authenticated users log in into AWS service through trusted authenticator like corporate LDAP or internet identity federated services. Where uses can access the account using temporary identity (STS = security token services) shared by the AWS service on behalf of 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.
- 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 a 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.
- Cross-account-policies using resource base policies: When users access a cross account resources using resource, base polices instead of role base polices – 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 assigned 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-credentails/ec2role
- IAM Role delegation: There are two polices 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:
- Resources based policies can be attached to the following AWS services ONLY
    - S3 Bucket
    - SQS
    - SNS
    - Glacier Vaults
- 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.

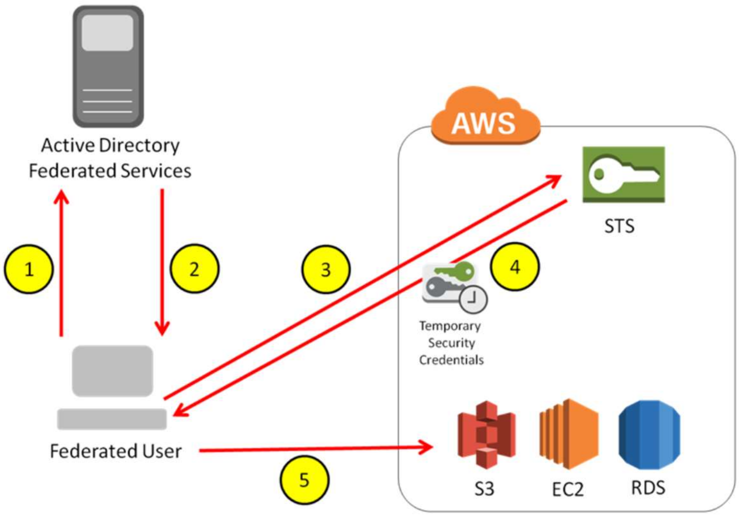
  <u>STS (Security Token Service)</u>

    - 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.
    - For those regions where STS is enabled by default, its
    - 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	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. It can be used by any IAM user or any user who have temporary access. This support MFA authentication. Max life span of the token is up-to 1 hrs. Minimum of 15 min
AssumeRoleWithSAML	This is used for establishing SAML 2.0 based federation between third party provided and the AWS IAM service. It can be used by any user that can pass an SAML authentication response that indicates authentication from a known identity provider. Max life span of the token is up-to 1 hrs. Minimum of 15 min
AssumeRoleWithWebIdentity	Any user that can send web authentication response that indicates authentication from a known identity provider. Max life span of the token is up-to 1 hrs. Minimum of 15 min

API call for STS	Description
GetSessionToken	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. Life span of the token is from 12Hr up-to 36 hours.
GetFedarationToken	This is used by IAM user or the root user. Life span of the token is from 12Hr up-to 36 hours.

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