**Software Development Kits (SDK) -**

IOS, Android, Browser (Java scripts), Java, .NET, Node.js, PHP, Python, Ruby, Go, C++

**Simple Queue Service (SQS)**

 - message oriented API

SQS - Message can contain upto 256KB of text, billed at 64KB chunks,

Single request can have 1 to 10 messages unto maximum of 256KB payload

Even though there is one message of 256Kb its basically 4 request for billing since (4 \* 64KB)

NO ORDER - SQS messages can be delivered multiple times in any order

Design - you can have 2 priority queues for priority based message one for higher and other for lower priority

EC2 instances always poll for messages from the queue (pull from the queue and not push)

Visibility timeout always start from when the application instance polled the message.

Great design - Visibility timeout expires that means there is a failure somewhere since that message was polled but not processed and hence not deleted so other some other process will poll the message again and visibility timeout starts again.

Visibility timeout by default is 30 Seconds up to 12 hour maximum (ChangeMessageVisibility) / maximum visibility

Maximum long polling timeout 20 seconds (http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-long-polling.html) —ReceiveMessageWaitTimeSeconds

Messages in the Queue can be retained for up to 14 days

First 1 million request ares free, then $0.50 PER EVERY MILLION REQUESTS

**Simple Notification Service (SNS)**

works on a publish - subscribe model, SNS notifies the message, and hence push based approach. Inexpensive pay as you go

CloudWatch or Autoscaling triggers SNS

SNS can notify to Email, Text / SMS, SQS or any HTTP end point.

protocols: HTTP, HTTPS, EMAIL, EMAIL-JSON, SQS or Application - messages can be customized for each protocol

SNS messages are stored redundantly to multiple AZs

SNS Dataformat - JSON (Subject, Message, TopicArn, MessageId, unsubscribeURL etc..)

$0.50 per 1 million SNS request

Different price for different recipient types

to HTTP: $0.06 / 100,000 notifications deliveries

to EMAIL: $2 / 100,000 notifications deliveries

to SMS: $0.75 / 100 notifications deliveries

http://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-devicetoken.html (CreatePlatformEndpoint API)

Message Example:

**SWF** - task oriented API

service makes it easy to co-ordinate work across distributed application components

Simple Work flow - human interaction to complete order or collection of services to complete a work order.

tasks --> individuals jobs assigned to workers

Workers 🡪 interact with SWF to get task, process received task and return the result

Deciders 🡪 program that co-ordinates the tasks, i.e. - ordering, concurrency (can be performed in parallel) and scheduling

Brokers --> programs that manages the interaction between workers and deciders. updates decider programs about the progress of each task. It ensures that task is assiged only once and never duplicated.

Domain (containers) --> isolates set of types (workflows), executions and task lists from others within the same account. This can be done at management console or using CLI with RegisterDomain parameter of SWF API

SWF maintains the application's state durably, workers and deciders run independently, and scale quickly

TASK is only assigned ONCE and NEVER DUPLICATED (key difference from SQS where messages can be processed multiple times)

SWF Maximum workflow processing time can be 1 year (equivalent seconds) - SQS is 12 hours processing time

**Shared Responsibility Model:**

1. for infrastructure services: (example, EC2, EBS, VPC)

AWS responsible for managing services that provided by AWS to user. like infrastructure level. (renting apartment)

User responsible for all the works that is done on top of these infrastructure.

2. for container services: (example, AWS RDS, AWS EMR, etc)

AWS manage --> underlying infrastructure and foundation service, operating system and application platform. for example, oracle, mysql

User manage --> customer data, firewall rules to access the data, configure data back-up and disaster recovery policies.

3. For abstracted Services: (example, S3, DynamoDB)

AWS manage --> underlying infrastructure and foundation service, operating system and application platform, most importantly IAM firewall security.

User Manage --> Customer data, encryption of data, access permissions using IAM (AWS managed) tool.

**CloudFormation 🡪** Control provision of resources as per need

Use of CFT, Beanstalk and Autoscaling are free but you pay for the AWS resources that these services create.

Fn::GetAtt - values that you can use to return result for an AWS created resource or used to display in output

By Default - rollback everything on error

Infrastructure as a code, Version controlled, declarative and flexible

"WaitCondition" parameter --> Stacks can wait based on this condition

**Intrinsic Functions 🡪** help to manage stacks. Helps to assign values available at runtime.

Limitation --> Currently, Intrinsic Functions can only be used in resource properties, outputs, metadata attributes, and update policy attributes. Can also be used to conditionally create stack resources.

Fn::FindInMap - returns the value corresponding to keys in a two-level map that is declared in the Mappings section.

Fn::GetAtt - returns the value of an attribute from the resources section of the template.

Fn::ImportValue - returns the value of an output exported by another stack.

Fn::Join - appends a set of values into a single value, separated by the specified delimiter. If a delimiter is the empty string, the set of values are concatenated with no delimiter.

Fn::Select - returns a single object from a list of objects by index.

Fn::Split – used to split a string into a list of string values so that you can select an element from the resulting string list

Fn::Sub - substitutes variables in an input string with values that you specify

Ref - returns the value of the specified parameter or resource.

* When you specify a parameter's logical name, it returns the value of the parameter.
* When you specify a resource's logical name, it returns a value that you can typically use to refer to that resource, such as a physical ID.

**Cloud Formation Templates 🡪 (JSON or YMAL)**

{

"AWSTemplateFormatVersion" : "*version date*",

"Description" : "*JSON string*",

"Metadata" : {

*template metadata*

},

"Parameters" : {

*set of parameters*

},

"Mappings" : {

*set of mappings*

},

"Conditions" : {

*set of conditions*

},

"Transform" : {

*set of transforms*

},

"Resources" : {

*set of resources*

},

"Outputs" : {

*set of outputs*

}

}

Description (optional) - A text string that describes the template. This section must always follow the template format version section.

Metadata (optional) - Objects that provide additional information about the template.

Parameters (optional) - Values to pass to your template at runtime (when you create or update a stack). You can refer to parameters from the Resources and Outputs sections of the template.

Mappings (optional) - A mapping of keys and associated values that you can use to specify conditional parameter values, similar to a lookup table. You can match a key to a corresponding value by using the Fn::FindInMap intrinsic function in the Resources and Outputs section.

Conditions (optional) - Conditions that control whether certain resources are created or whether certain resource properties are assigned a value during stack creation or update. For example, you could conditionally create a resource that depends on whether the stack is for a production or test environment.

Transform (optional) - For serverless applications (also referred to as Lambda-based applications), specifies the version of the AWS Serverless Application Model (AWS SAM) to use. When you specify a transform, you can use AWS SAM syntax to declare resources in your template. The model defines the syntax that you can use and how it is processed.

Resources (required) - Specifies the stack resources and their properties, such as an Amazon Elastic Compute Cloud instance or an Amazon Simple Storage Service bucket. You can refer to resources in the Resources and Outputs sections of the template.

Outputs (optional) - Describes the values that are returned whenever you view your stack's properties. For example, you can declare an output for an S3 bucket name and then call the aws cloudformation describe-stacks AWS CLI command to view the name.

**LAMBDA**

**Lambda Exam Tips:-**

1. Lambda scales out (not up) automatically.

2. Lambda functions are independent, 1 event = 1 function

3. Lambda functions can trigger other lambda functions, 1 event can = X lambda functions if functions trigger other functions

4. Lambda is serverless. Its good to know what services are serverless in AWS, like DynamorDB, API Gateway, etc.

5. AWS X-Ray service, can allow you to debug code what is happening. Very helpful incase of Architecture gets complicated.

6. Lambda can operate globally, you can do things like back-up of S3 bucket to another bucket

7. its good to know what services can triggers Lambda functions.

**ElasticBeanstalk**

Its uses ASG,ELB,EC2,RDS,SNS and S3 to provision things.

to quickly deploy, scale, and monitor applications, with the ability to provision infrasture as per application need. Just upload the application code and AWS BeanStalk will provision the resources.

>> Any updates to application can be done 1 instance at a time, or % percentage of instances, immutable updates (which means, all updates will launch new instance and once updates successful to new instance, it will terminate previous instances)

>> if RDS instance created by ElasticBeanstalk, it will be deleted when user delete application. But if you created RDS separately. And using that RDS instance as part of ElasticBeanstalk application, then RDS instance will stay even if you delete application.

Environment Tier - Webserver, Worker

Predefined Configurations - IIS, Node.JS, PHP, Python, Ruby, Tomcat, Go, .NET, JAVA, packer

preconfigured docker: Glassfish, Python or generic docker

Environment URL - has to be unique

Dashboard - Recent events, Monitor, Logs, Alarms, Upload and Deploy and Configurations

Configuration - Scaling, Instances (DIRTMCG instance types, key pair), Notifications, Software configuration (e.g. PHP.ini), Networking tier (ELB, VPC config), Data tier(RDS)

Environment properties (Access key and secret key as parameters)

Elastic Cache 🡪 cache retrieval is faster than disk-based databases.

It supports 2 in-memory cache engines:

1. Memcached

2. Redis

**RedShift 🡪** OLAP (AWS Data Warehouse solution)

**RDS 🡪** OLTP (mostly)

available in AWS: MySQL, PostgreSQL, MS SQL Server, Aurora, MariaDB, Oracle

**DynamoDB**

fast - flexible No sql database - single digit ms latency, fully managed, supports document and key-value (web, gaming, ad-tech, IOT)..

Table, Item (row), attribute (key - value)

Eventual Consistent Reads vs Strongly Consistent Reads

Read Capacity Units, Write Capacity Units (can be scaled up) - push button scalability

Writes are written to 3 different location / facilities/ datacenter (synchronous) - Amazon DynamoDB synchronously replicates data across three facilities in an AWS Region, giving you high availability and data durability.

Two types of primary key -

(1) Single Attribute (think unique id) - Partition Key (Hash Key) composted of 1 attribute (no nesting allowed here) - Partition key will help determine the physical location of data.

(2) Composite key (think unique id and range) - Partition Key(Hash Key) & Sort Key (Range key - e.g date) - composed of 2 attributes - if two data have same partition key (same location) it must have a different sort key, and they will be stored together on single location.

Secondary Indexes

(1) Local Secondary Index - Same Partion Key + Different Sort Key ( can only be created while creating the table, cannot be added/removed or modified later)

(2) Global Secondary Index - Different Partition Key + Different Sort Key ( can be created during the table creation or can be added later or removed / modified later)

DynamoDB Streams

use to capture any kinda modification to the dynamo db table, Lambda can capture events and push notifications thru SES

Table can be exported to csv (either select all items )

Query vs Scan

Query operation finds item in a table using only primary key attribute values , must provide partition attribute name and the value to search for, you can optionally provide a sort key attribute name and value to refine search results (e.g. all the forums with this ID since last 7 days). By default Query returns all the data attributes for those items with specified primary keys. You can further use ProjectionExpression parameter to only return a selected attributes.

Query results are always sorted by the sort key (ascending for both numbers and string by default). To reverse the sort order set the ScanIndexForward parameter to false

By Default Queries are going to be Eventually consistent but can be changed to StronglyConsistent.

Scan operation is basically examines every item - e.g. dumping the entire table, by default Scan returns all the data attributes but we could use ProjectionExpression parameter to only return a selected attributes.

Query operation is more efficient than scan operation

For quick response time, design your table in a way that you can use Query Get or BatchGetItem API (read multiple items - can get up to 100 items or up to 1MB of data) ,

Alternatively design your application to use scan operation in a way that minimize impact of your table’s request rate since it can use up the entire table’s provisioned throughput in a single scan operation

**DynamoDB Provisioned Throughput calculations**

Items == rows

Read Provisioned Throughput

* All units are rounded up to 4KB increments
* Eventual Consistent reads (default) consist of 2 reads per second
* Strongly Consistent reads consist of 1 read per second

( Size of Read Rounded to nearest 4KB Chunk / 4 KB \* no of items ) / 2 <— if eventual consistency

( Size of Read Rounded to nearest 4KB Chunk / 4 KB \* no of items ) / 1 <— if strongly consistency

Write Provisioned Throughput

* All units are rounded up to 1KB increments
* All writes consist of 1 write per second

( Size of write in KB \* no of items ) / 1

When you exceed your maximum allowed provisioned throughput for a table or one or more global secondary index you will get 400 HTTP Status code - ProvisionedThroughputExceededException

AssumeRolewithWebIdentity role

Idempotent = conditional write

Atomic counters - always need to increment so its not idempotent

if data is critical and no margin of error, then must use Idempotent conditional write.

http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Limits.html#limits-tables

Only Tables (256 table per region) and ProvisionedThroughput (80 K read, 80K write per account for US east, 20K for other regions) limits can be increased

http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/QueryAndScanGuidelines.html (Reduce Page Size for Scan operation and Isolate Scan Operation)

**Simple Storage Service (S3)**

* secure, durable, highly scalable object store (0 bytes to 5TB), universal namespace (must be unique bucket - regardless of regions),object based key value store, VersionID, Metadata, ACL
* The total volume of data and number of objects you can store are unlimited. Individual Amazon S3 objects can range in size from 0 bytes to 5 terabytes. The largest object that can be uploaded in a single PUT is 5 gigabytes. For objects larger than 100 megabytes, customers should consider using the Multipart Upload capability. it mean the largest single file into S3 is 5G, but after the 5G files are in S3, they can be assembled into a 5T file,
* You can use a Multipart Upload for objects from 5 MB to 5 TB in size (Exam question, scenario where more than 5GB file needs to be uploaded)
* object based storage vs block based Storage (EFS)
* data is spread out in multiple facilities, you can lose two facilities and still have access to files
* For PUTS of New Objects (Read after Write Consistency), For Overwrite PUTS (update) and DELETE (Eventual Consistency)
* Updates in S3 are atomic updates. means update anything on S3, will either get new data or old data, never gets partially updated data.
* http://docs.aws.amazon.com/general/latest/gr/aws*service*limits.html#limits\_s3 ( Number of S3 bucket limit per account — 100)

Storage Tiers/ Class

S3 Standard - Durability (11 9s), Availability (99.99 %) - reliable regular for just about everything

S3 IA (Infrequent Access) - Durability (11 9s), Availability (99.9 %) - accessed every 1 month to 6 months or so (infrequent) but rapid access and low retrieval time (few ms)

S3 RRS(Reduced Redundant Storage)- Durability (99.99%), Availability (99.99 %) - less durability (data that can easily be regenerated - e.g thumbnails) - cheapest of all s3, less fault tolerant then the other two since you are willing to loose the data, reproducible data

Glacier - for archival only (3 to 5 hours restore time)

S3 price - charged for Storage, number of requests, data transfer (tiered so more you use less charge)

bucket name has to be all lowercase letters

S3 for static website hosting (Static Website Hosting > Enable website hosting) - no dynamic

Any time you create a bucket nothing is publicly accessible / Any time you add an object to a bucket its private by default (you will get 403) > Make the files public (even for public hosting)

every object inside the bucket can have different storage class (S3 standard, S3-IA, S3-RRS) and you can turn on server side encryption (AES - 256)

regular bucket link: https://s3-eu-west-1.amazonaws.com/ankittest <— https

bucket with Static website hosting: http://ankittestsite.s3-website-eu-west-1.amazonaws.com <— http (has to be for static hosting), you can turn it into SSL / https with cloudfront though

CORS (Cross Origin Resource Sharing) - to avoid the use of proxy

Enable one Domain Name (DNS) trying to interact with another Domain Name (DNS)

Versioning - once enable you cannot disable versioning / although it can be suspend it , if you want to turn it off delete the bucket and recreate (version id)

Once you delete the delete marker, you can get the file back that you have deleted while versioning on

every version is stored separately in the bucket for each version / might not be a good choice for cost perspective for large media files., multiple updates use case also not ideal for versioning.

Versioning’s MFA Delete Capability can be used to provide additional layer of security.

Cross Region Replication - (requires versioning enabled on source and destination buckets)

you can enable - need source and destination bucket (create a new bucket, source bucket will not show up on drop down of destination)

Existing objects will not be replicated, only new objects will be replicated across the region

Lifecycle management in S3

(1) when versioning is disabled

Transition to IA S3 - min 30 days and has a 128KB minimum of object size

Archive to Glacier - min 1 day if IA is not checked, min 60 day if Transition to IA S3 is checked

Permanently Delete - min 2 day if IA is not checked and 1 is selected for Glacier, min 61 day if IA is selected 30, Glacier is selected 60.

(2) when versioning is enabled you have lifecycle management options to take action on previous version as well as current version.

Security and Encryption in S3

* by default newly created buckets are private
* Access control using Bucket Policies (entire bucket) and ACL(individual objects and folders)
* access logs - all the request made to S3 buckets, to another bucket or another account’s S3 bucket

Encryption

(1) In Transit - SSL / TLS

(2) Data at rest

Server Side Encryption

* SSE- S3 Server Side Encryption with S3 managed keys, (amazon AES 256 handled for you) - click on the object and encrypt
  + each object is encrypted with unique key
  + additionally, Amazon encrypt the key with a master key, which amazon rotates regularly.
* SSE - KMS - AWS Key management services , managed keys - additional charges / audit trail of keys, amazon manage keys
  + envelop key that protects your data's encryption key
* SSE - C - Server side encryption with Customer provided keys

Client Side Encryption

* you encrypt the data on client side and upload to s3

Every non-anonymous request to S3 must contain authentication information to establish the identity of the principal making the request. In REST, this is done by first putting the headers in a canonical format, then signing the headers using your AWS Secret Access Key.

You can insert a presigned url into a webpage to download private data directly from S3.

The object creation REST APIs (see Specifying Server-Side Encryption Using the REST API) provide a request header, x-amz-server-side-encryption that you can use to request server-side encryption.

S3 Transfer Acceleration

Utilize local edge locations to upload content to S3 - incur extra cost

further away you are the more benefit you get (faster)

can use different URL to transfer data to Edge location and then back-up to S3.

URL could be like 🡪 bucklet-name.s3-accelerate.amazonaws.com

S3 Storage Gateway

used for back-up purposes, data first go to storage gateway and then it’s being replicated to all S3

File Interface 🡪

* NFS (Network File System Protocol) --> this is used to access files over a network in a secure way.
* unlimited storage, although file size should not exceed 5 TB.
* uploading files first using NFS and then AWS will replicate that to S3 using some encryption

Volume Interface 🡪

(1) Gateway stored volumes - entire dataset is stored onsite and asynchronously backed up to S3

(2) Gateway cached volumes - Most frequently used data is stored onsite and entire dataset is stored on S3

Tape Interface 🡪

Gateway Virtual Taped library - Used for backup if you don’t want to use Tapes, like Netbackups etc..

Virtual Tape Library --> backed-up by Amazon S3. Access is immidiate.

Virtual Tape Shelf --> backed-up by amazon Glacier. Need some hours to access these.

**Snowball (Import Export)** -- Previously, snowball is known as import/export service

designed for high volume of data transfer to AWS

1. Standard Snowball 🡪 import to S3 and export from S3. store upto 80 TB only. No compute capacity.
2. Snowball Edge 🡪 (Import to S3, EBS, Glacier and export from S3) can store upto 100 TB, with high compute capacity. It can be clustered as well
3. Snowmobile 🡪 move upto 1 XB exa bytes of data (100 PB), huge compute and storage capacity

S3 stored data in alphabetical / lexigraphical order. so if you want to spread the load across S3, filename should not be similar (Optimize performance)

**CloudFront**

Content Delivery Network - edge locations, reduced latency, traffic serves from the closest nodes

CloudFront can be used for both Download and upload files.

Edge locations - content will be cached (over 50), different from region / AZ. TTL (speed of image // media is quicker - first user suffers the performance), can be not only read only (you can write it)

Origin can be - S3, EC2, ELB, Route53 also NON AWS origin server ,

Distribution - name given to the CDN consist of collection of Edge locations

(1) Web Distribution -

(2) RTMP (media streaming / flash) Distribution - for Adobe flash files only

you can have multiple origins of a distribution

Path Pattern (\*)

Restrict viewer access by signed URL or Signed Cookies

Restrict content based on geo location (whitelist {allowed countries} and blacklist {restricted countries})

Invalidation basically removes the objects from Edge location cache memory, but it cost money. This is needed when you wants to remove objects before TTL time.

Create invalidate - invalidate TTL (you pay for it) like purge in Akamai

**VPC** - logical datacenters in AWS

Can span multiple AZ, but can’t span multiple regions, PEER VPC, but no Transitive Peering

Custom VPC has to be /16 can’t go higher then that /8 is not allowed

When you create Custom VPC it creates default security group, default network ACL and default route table., it doesn’t create default Subnet

One Subnet == one AZ, you can have security group spanning multiple AZ, ACL’s span across AZ (assign sg and ACL to two different subnets)

security groups are attached to one VPC. security groups can't span across multiple VPCs.

any CIDR block 5 reserved IPs (.0, .1, .2, .3, .255)

so for CIRD block /24: 2^8 - 5 = 256 - 5 = 251 available IP address space

when you create internet gateway, by default its detached, attach it to VPC then, **only 1 IGW per VPC**

When you create a VPC Default Routetable(Main Routable) is created where the default Routes are,

10.0.0.0/16 Local <— all subnets inside VPC will be able to talk to each other

Don’t touch Main route table

Create another routetable for route out to internet (0.0.0.0/0 IGW) <— route out to the internet

Last thing you associate this new route table to one of the subnet which will make it public. (you can enable auto assign public IP for the public subnet)

1 subnet can have 1 routetable

ICMP is for ping / monitor

VPC peering can be done using another AWS account or with other VPCs in same account.

Route Table --> specifies how packts are forwarded between the subnets within your VPC, internet and VPN connection.

Internet Gateway --> is a virtual router that connects a VPC to the internet.

NAT instance and NAT gateway (network address translation)

NAT Instance - disable source / destination check., always behind security group, must be in public subnet, must have an EIP, must be a route out of the private subnet to NAT

Increase the instance size if bottleneck

Change the main route table - add a route (0.0.0.0/0 NAT Instance target)

NAT Instance is a single point of failover (put it behind a ASG), each NAT instance specific to one AZ.

NAT Gateway are also specific to each AZ, but when you configure your route table, failure handle can be done automatically.

NAT Instance can’t scale automatically, hence NAT Gateway is introduced. But Bastion server can scale automatically.

NAT gateway - released in 2016 - amazon handled

* Amazon maintains it for you, no need to handle yourself. (security patches applied by AWS)
* You can just create the gateway and assign EIP (put it in public subnet) (automatically assigned)
* Change the main route table - add a route (0.0.0.0/0 NAT gateway target)
* No need for disable source/destination check or no need to put it behind a security group - it handles it for you.
* Highly available / redundancy no need for ASG. NAT gateways are little bit costly - always use it in production scale automatically up to 10Gbps

NAT instance supports Bastion servers but NAT Gateway don’t support.

Bastion Server (JumpBox) 🡪 used for administrative purpose, to access private subnet through internet

Public IP gets changed every time for an instance after stop/start, Elastic IP address is static, doesn’t change after stop/start.

**NACL vs SG**

Security groups are stateful - any inbound rule , applies to outbound as well (Only Allow rules)

* by default all inbound deny, all outbound allow
* can span across AZ

ACL are stateless -

* For default ACL, all inbound and outbound rules are allowed by default - associated with all subnets in VPC by default
* for Custom ACL, all inbound and outbound traffic is denied by default - not associated with any subnet
* 1 subnet is only associated with ACL. granular rules for ACLs, numbered rules (recommended steps of 100)
* rule number 99 takes precedence over rule number 100 (if 99 is blocked and 100 is allowed) 99 will be executed.
* Can SPAN across AZ

Ephemeral port - 1024 - 65535 should be allowed to take traffic.

If you want to BLOCK IP address then must use ACL, because security group doesn’t have deny

Bastion - keep it in public subnet to allow SSH / RDP into instances into private subnets (High availability - Bastion in two public subnets and also ASG - Route 53 running Health checks on those Bastion).

Bastion is used to securely administer EC2 instances in private subnet through internet access (using SSH or RDP)

VPC Flowlogs: to capture all the traffic information into logs - logs everything (create IAM role and create cloud watch log group - and log stream)

1. Peer VPC flow logs can't be enable, unless it is in same account

2. Currently, Tagging in flow log is not available.

3. Flow log configuration can't be changed after creation.

VPC Cleanup: can’t delete VPC if you have active running instance or ELB is running

VPC End Points: allows user to connect VPC with other services. EndPoint works on private IP addresses (not on public IP address), as it is a connection within cloud network or private network.

Interface EndPoint --> Entry point for traffic destined for a service. This works for specific service by service basis, not highly available. its an Elastic Network Interface.

Gateway EndPoint --> target for a route in your route table for traffic destined for a service. Its highly available and available for different services.

**EC2 -** Resizable compute capacity, scale up or down in minutes or instantly.

EC2 Pricing model

On Demand --> linux (per sec) and Windows (per hour)

Reserved Instances (RI) --> 1 to 3 yr basis, all upfront, partial upfront, no upfront

types:-

1. Standard RI --> upto 75% sale

2. Convertible RI's --> upto 54% sale, allowed to change configuration of RI

3. Scheduled RI's --> used at specific time frame

Spot --> rate based on the availability (market demands) at any specific time in specific region. (user bid for EC2 prices)

>> bid price must be more than spot price, hence user will get that instance. If spot price goes up then the bid price

then AWS terminates the instance.

> If you terminate the instance before allocated time you will pay for that time duration either way (for example, that hour)

> If AWS terminate the instance before the allocated time because spot price goes above the bid price,

then you get the hour it was terminated in for free.

> it is used when application is flexible with start and end time.

Dedicated Hosts --> no shared resources, yearly or longer duration plan.

**Different EC2 Instance types:-**

**DR Mc GIFT PX** ------>

D --> Dense Storage (MPP Data Warehouse, HDD, High Disk throughput)

R --> Memory (RAM) optimize

M --> (Main / default) General purpose

C --> compute optimize

G --> Graphics Intensive (Game/3D) -- Accelerated compute optimize

I --> I/Ops (High I/O speed storage, SSD)

F --> Field Programmable Gate Array (Hardware Acceleration for your code)

T --> Low cost , General purpose

P --> Graphics, General purpose (ML, Blockchain) -- Accelerated compute optimize

X --> extreme memory optimized (SAP/Spark)

H --> High Disk throughput (I/O Storage Optimized, HDD)

**EBS (Elastic Block Size) --** Disk on top of CPU (HDD or SSD)

Purpose 🡪 create storage volume (EBS) and attach to EC2 instances as per needed data storage purpose.

EBS are usually placed in AZ and replication and security is guaranteed.

Can’t mount one EBS volume to multiple EC2 instances, instead use EFS.

only bootable volumes (GP2, IO1, magnetic HDD) can be assigned as Root volume.

1 subnet = 1 availability zone

By default delete the volume on instance termination, need to uncheck this option to avoid.

Termination protection is turned off by default, you must turn it on.

Root volume of EC2 instance can't be encrypted directly, need to take snapshot and encrypt. But any other added volume can be encrypted by checkbox selection.

For windows AMI need to use third party tools for encryption

**EC2 important keys**

1. public key: allocated by AWS. User can have one public key across all EC2 instances. but can create multiple private key.

2. private key: stored by individual user. only one private key per user for specific EC2 instances.

**\*\* EBS Volume Types:**

1. General Purpose SSD (GP2) --> solid state drive, bootable

> balance in price and performance

> upto 10000 IOPS

2. Provisioned IOPs SSD (IO1) 🡪 bootable

> designed for I/O intensive applications such as large RDBMS, NoSQL

> up to 20000 IOPS

Magnetic storage Volume (Hard disk drive)

3. Throughput Optimized HDD (ST1) --> for large data size, not- bootable

> can't be bootable

> sequential read/write

> frequently access workload

4. Cold HDD (SC1) 🡪 not- bootable

> Low cost storage

> file server

> can’t be bootable

5. Magnetic (Standard) --> same as cold but bootable

> lowest cost volumes that is bootable

1. System status check: to check instance is reachable

2. Instance Status check: to check operating system installed on that instance (server) accepting traffic.

Security Groups --> AWS virtual firewall

1. define Inbound / Outbound ports that are allowed, cant specify deny rules)

2. Any rule apply to inbound automatically apply to outbound. This is called stateFul.

Hence, security groups are STATEFUL.

But Newtork Access Control Groups are STATELESS, because Inbound and outbound rules are seperate.

3. By default, all inbound traffic is denied, but all outbound traffic is allowed.

specifically mention traffics that we allow for inbound.

can't specify deny rules

4. one security groups --> can have many EC2 instances

5. multiple security groups attached to an EC2 instances.

**Volumes and Snapshots:**

Volumes exsits on EBS, and its a virtual hard disk.

Snapshots: exists on S3. Its nothing but point in time copy of Volumes.

Snapshots are incremental. means, next time when you take snapshot it will copy only blocks of volume that has chnaged.

1. you can create AMI's from both volumes and snaps.

2. you can change volume size and storage type on the fly, while making copy.

3. Volume should always be in same AZ where EC2 instance is.

4. To move one EC2 instance from one AZ/Region to another, need to create snap or AMI, then copy it to new AZ/Region.

**Volumes and Snapshots security** -- (this will bring 4-5 questions in exam)

1. snapshots of encrypted volumes are encrypted automatically.

2. volumes restored from encrypted snapshots are also encrypted automatically.

3. you can share snapshots but only if they are unencrypted. because encryption key is tied to your account.

4. These snapshots can be shared as public or with other AWS accounts.

EC2 Storage or Root Device Volume: have 2 types

1. Instance Store (Ephemeral Storage)

2. EBS Backed Volumes

**EBS vs Instance Store type:**

1. Instance Store volume is known as Ephemeral storage

2. Instance Store volumes can not be stopped. If underlying host fails, you will loose data.

3. EBS backed instances, can be stopped. you will not loose your data while doing this.

4. you can reboot both EBS and Instance Store and will not loose your data.

5. By default root volumes will be deleted, on termination. However with EBS volumes, we can configure AWS to keep the EBS volume

on instnace termination.

6. While launching EC2 instance user can attach more Instance Store volumes but not after launching EC2 instnace. On the contrary, user can attach EBS volume while launching as well as after launching EC2 instance.

7. Also, Instance Store volumes can be encrypted.

8. User can't even detach Instnace store volume but in case of EBS volume, user can detach from one EC2 and attach to another EC2.

**AMI Exam Tips:**

AMI's are regional.

you can only launch AMI from the region it is stored.

However you can copy AMI from one region to other.

ECS stands for Elastic Container Service. It manages running containers on your EC2 instances. It does not act as a scheduler,

and it is neither server-less nor software that you manage.

**EC2 Instance metadata and user data** 🡪

Q: How you will get public-ip address while you logedin to EC2 through CLI.

A: you need to look to meta-data of EC2 using above command and not the user-data.

**EFS (Elastic File system)**

data is stored across multiple AZs within a region

read after write consistency

Multiple EC2 can connect to one EFS but one EC2 can connect to only one EBS. EFS is storing files centrally.

access restrictions to directory and file level.

**Load Balancing and Auto Scaling Groups** -->

1. Auto Scaling Group --> define rules, how to scale up or down EC2 instances

for example:, based on CPU utilization lauch 1 EC2 instance

>>if CPU utilization consistently 90% for more than 5 min launch 1 more EC2 instance

>> if CPU utilization less than 20%, down grade (terminate) one instance, etc...

**EC2 Placement Groups** --> logical groups of instances within Single AZ

Benefits --> gain high network throughput,

unique name within AWS account,

Limitation --> Can't span across multiple AZ

supported instance type (compute / storage optimized, GPU, memory optimized)

AWS recommend Homogeneous instances within one placement group

can't merge a placement groups

can't move existing instance into placement group. Need to create AMI and launch new instance to placement group

**ELB (Elastic Load Balancer)**

Functionalities:-

distributes incoming application traffic

consistent health check of EC2 instance

ELB only have DNS url, no public IP address.

Instances monitored by ELB are reported as:: InService or OutOfService

**type of ELB:**

1. Application Load Balancer (HTTP / HTTPs)

2. Classic Load Balancer (previous generation for HTTP, HTTPs, TCP, SSL)

3. Network Load Balancer (TCP)

**CloudWatch**

create events or alarm for any event alerts.

two categoris of metrics --> 1. By Auto Scaling Group 2. Per-Instance Metrics

under these categories 4-5 different types are available:

>> Network related

>> StatusCheck Related

>> Disk Related

>> CPU Related

>> All other are custom metric, like memory related. Need to write some code for it.

**CloudWatch Exam Tips -->**

1. Standard Monitoring = 5 mins

2. Detailed Monitoring = 1 min

**CloudWatch and CloudTrail difference -->**

CloudWatch --> used to log or monitor activities within AWS services

CloudTrail --> used for Audit purposes of AWS account. Used to log activities with AWS services. Example, creating new IAM user, S3 bucket, etc.

**IAM (Identity and Access Management)**

IAM is not Region specific. Its a global/universal.

>> allows to setup our own password rotation policies / rules.

User 🡪 It’s an identity, to access AWS resources.

* Every user assigns an Access Key ID and secret Access key to access AWS through CLI or SDK.
* Users userid and password is used to access AWS consol.
* By default users don't have access to AWS resources.
* it’s possible to assign specific policy to users
* to add user to a group
* to copy permissions from other users account

Groups 🡪

* Collection of users, on which we can allocate collection of policies at one shot.
* group is not truly an identity, It is only a way to attach policies to multiple users at one time

Roles 🡪

* similar to IAM users, it is an identity, with Collection of policies bundled together as per need.
* choose AWS service that can use that specific role.
* roles does not have any credentials (password or access keys) associated with.
* roles can be assigned to a federated user
* IAM roles issue keys that are valid for short durations, making them a more secure way to grant access.
* However, you don't sign in to a role, but once signed in as a user you can switch to a role. users can have either permissions associted with Role or Users own permissions but can't have both at same time.
* The role can be in your own account or any other AWS account.
* By default, your role session lasts for one hour. but this can be increased or decreased.
* If you use role chaining, your session duration is limited to a maximum of one hour, operation fails beyond that.
* Roles are more secure than storing your access key and secrete key
* Roles are universal, can be used in any region.

Temporary Credentials:

* have a more restricted set of permissions than your standard IAM user.
* A benefit is that they expire automatically after a set period of time and you can set this duration.

Policy Documents:

* A JSON document that defines one or more permissions.
* It is universal and can be applied directly to users, groups and Roles.

Security Token Service (STS) --->

Federation / identity store 🡪 like (facebook, google, active directory).

Identity Broker:

>> service that allows you to take an identity from one domain and join or use it to another domain.

>> User create their own Identity broker most of the time.

>> It mostly validate username and password from the specific domain from LDAP directory

>> Identity Broker than calls GetFederationToken function, (which pass/input IAM policy and duration) to get permission to temporary security credentials for that IAM role.

>> STS confirms that the IAM user with valid policy making a call to GetFederationToken function and will response following:

access key, secret access key, token, and duration (the token's lifetime) as per request. These details then can be used to

access AWS resources.

>> Identity Broker sends above information to application that is making call.

Identity Broker -- Exam tips with 2 different scenarios:

scenario 1:-

>> Develop Identity broker to communicate with LDAP and AWS STS

>> Identity Broker always authenticate with LDAP first, then with AWS STS

>> Application then gets temporary access to AWS resources

scenario 2:-

>> Develop Identity broker to communicate with LDAP and AWS STS

>> Identity Broker always authenticate with LDAP first, gets an IAM role associated with IAM user

>> application then authenticate with STS and assume that IAM role

>> Application then uses that IAM role and gets temporary access to AWS resources

Q: Can you authenticate using AD and What type of Authetication would to do so??

A: Yes, using SAML (Security Assertion Mark-up Language). SAML basically provides temporary security credential to sign-on to AWS.

Need to check this --> AssumeRoleWithSAML is the parameter that provides these credentials.

**Route53**

Domain Name Services -- convert human friendly name (acloud.guru) into internet protocol Addresses

* Route53 is not free a service.
* Route53 is global service.
* “A Record” translates the domain name to IP address.
* CNAME --> canonical Name, is used to resolve one domain Name with another.
* Alias Record --> to resolve naked domain name (zone APEX address) to ELB's DNS address.
* Naked Domain Name --> domain name that doesn't contain "www" or any other initial letters before it.

Important to remember -->

1. ELB never uses IP addresses although it always uses Domain Name.

2. Difference between Alias record and CNAME

3. Always choose Alias record over CNAME.

Answer --> why?? because Alias record are free, Alias record allows you to map Naked domain name with ELB's DNS name.

Different Routing policies:--

Simple, Weighted, Latency, Failover, GeoLocation

**Error Codes**

HTTP 3XX – Redirection

HTTP 4XX - Client Side Error

HTTP 403 – Access Denied

HTTP 404 – NoSuchBucket, NoSuchKey, NoSuchLifecycleConfiguraion, NoSuchUpload, NoSuchVersion

HTTP 409 – BucketNotEmpty (The bucket you tried to delete is not empty)

HTTP 5XX - Server Side Error