***DevOps Associate Certification Notes***

**Identity Access Management**

* IAM is universal and not region specific
* IAM consists of the following:
  + Users
  + Groups (A way to group users and apply policies to them collectively)
  + Roles
  + Policy Documents: Made out of JSON
* The root account is simply the account created when first setup your AWS account. It has complete Admin access
* New Users have NO permissions when first created
* New Users are assigned **Access Key ID & Secret Access Keys** when first created
* These are not the same as a password, and you cannot use the Access Key ID & Secret Access Key to login in to the console. You can use this to access AWS via the APIs and Command line however:
* You only get to view these once. If you lose them you have to regenerate them
* Always setup Multifactor Authentication on your root account
* You can create and customize your own password rotation policies
* Can we authenticate with ActiveDirectory? Yes and is using SAML
* It authenticates with ActiveDirectory first and then a temporary security credential is assigned
* You can authenticate using applications such as Facebook, Google, etc (Look Web Identity Federation with Mobile Applications)
  + The process is done by web app granting a token
  + You get an RoleArn (Amazon Resource Name)
  + Once you get your credentials you call AssumeRoleWithWebIdentity to access all the Amazon resources available to you

**Elastic Cloud Compute (EC2)**

* EC2 Options:
  + **On Demand**: Allows you to pay a fixed rate by the hour with no commitment
  + **Reserved**: provide you with a capacity reservation, and offer a significant discount on the hourly charge for an instance. 1 year or 3 year terms
  + **Spot**: enable you to bid whatever price you want for an instance capacity, providing for even greater savings if your applications have flexible start and end times
* When to use On Demand:
  + Users that want the low cost and flexibility of Amazon EC2 without any up-front payment or long-term commitment
  + Applications with short term, spiky, or unpredictable workloads that cannot be interrupted
  + Applications being developed or tested on Amazon EC2 for the first time
* When to use Reserved:
  + Applications with steady state or predictable usage
  + Applications that require reserved capacity
  + Users able to make upfront payments to reduce their total computing costs even further
* When to use Sotp
  + Applications that have flexible start and end times
  + Applications that are only feasible at very low compute prices
  + Users with urgent computing needs for large amounts of additional capacity
* If the Spot instance is terminated by Amazon EC2, you will not be charged for a partial hour of usage. However, if you terminate the instance yourself, you will be charged by any hour in which the instance ran
* EC2 Instance Types:
  + **D** for Density (Dense Storage)
  + **I** for IOPS (High Speed Storage)
  + **R** for RAM (Memory Intensive)
  + **T** cheap general purpose (Lowest cost, general purpose)
  + M Main choice for general purpose apps
  + **C** for Compute (Compute optimized)
  + **G** for Graphics (Video encoding, machine learning, 3D application streaming)
* You can attach multiple Elastic Block Source (EBS) to a single EC2, but you CANNOT attach one EBS to multiple EC2 instances
* EBS Volume Types
  + General Purpose SSD (GP2)
    - Designed for 99.999% availability
    - Ratio of 3 Input Output Per Second (IOPS) per GB with up to 10,000 IOPS and the ability to burst up to 3000 IOPS for short periods of volumes under 1Gib
  + Provisioned IOPS SSD (IO1)
    - Designed for I/O intensive applications such as large relational or NoSQL databases. Use if you need more than 10,000 IOPS
  + Magnetic (Standard)
    - Lowest cost per gigabyte of all EBS volume types. Magnetic volumes are ideal for workloads where data is accessed infrequently, and applications where the lowest storage cost is important
* By default the volume is deleted when the EC2 instance is terminated. This is determined by a checkbox when creating the volume.
* The volume is not encrypted and AWS does not offer another option where the operating system goes. If you add another volume you can set it to be encrypted or not
* Termination Protection is off by default, you must turn it on
* On an EBS-backed instance, the default action is for the root EBS volume to be deleted when the instance is terminated
* Need to avoid User Access Key ID and Secret Access Key, the more secure easier way to handle access is to use roles
* Always use Roles, is much more secure

**Instance MetaData**

* To get The instance MetaData you need to do the following:
  + Login into your EC2 instance
  + Sudo to become root
  + The IP Address to get all the options is <http://169.254.169.254/latest/meta-data>
  + If you type curl <http://169.254.169.254/latest/meta-data>/ you will get all the different options
  + If you type curl <http://169.254.169.254/latest/meta-data>/public-ipv4 it shows the ip address of the EC2 instance

**Elastic Load Balance**

* ELB’s are not free, you are charged by the hour and on a per Gb basis of usage
* Service which are free include CloudFormation, Elastic Beanstalk, Autoscaling, Opsworks
* The service that creates resources are free the actual resource is not
* ELB Listener
  + Before you start using ELB, you must configure one or more listeners for your load balancer
  + A listener is a process that checks for connection requests to your load balancer
  + It is configure with a protocol and a port for front-end (client to load balancer) connections, and a protocol and a port for back-end (load-balancer to back-end instance) connections
* Suported ELB Protocols
  + HTTP
  + HTTPS
  + TCP
  + SSL (secure TCP)
* ELB supports all the ports 1 - 65535
* HTTP Error Codes Exam Tips
  + 200 - The request has succeeded
  + 3xxx - Redirection
  + 4xx - Client Error (think 4040 not found)
  + 5xx - Server Error

Know the available SDKs

* To find them go to <https://aws.amazon.com/tools>
  + Android, iOS, JavaScript (Browser)
  + Java
  + .Net
  + Node.js
  + PHP
  + Python
  + Ruby
  + Go
  + C++
* The default region is US-EAST-1 (Java has default region, Node.js does not)

**Simple Storage Service (S3)**

* S3 stores data in alphabetical order
  + To optimize S3, make sure that if you are loading a lot of files the beginning of the names are different so it gets spread out rather than inserting it into a single bucket
* The largest size on a PUT operation is 5GB
* S3 us Object based i.e allows you to upload files
  + Cannot install an Operating System on S3
* Files can be from 1 Byte to 5TB
* There is unlimited storage
* Files are stored in Buckets
* S3 bucket is a universal namespace, that is, names must be unique globally
* Namespace:
  + https://s3-<region>.amazon.com/<bucket name>
* New objects written to S3 are available immediately (Read after Write consistency for PUTS of new Objects)
* Updating and/or deleting an object can take some time to propagate (Eventual Consistency for overwrite PUTS and DELETES (can take some time to propagate))
* S3 is a simple key → value store
* S3 objects consists of the following:
  + **Key**: this is simply the name of the object
  + **Value**: this is simply the data and is made up of a sequence of bytes
  + **Version ID**: important for versioning
  + **Metadata**: data about the data you are storing
  + **Subresources**:
  + **Access Control Lists**
* Amazon guarantees 99.99% availability for the S3 platform
* Amazon guarantees 99.999999999% durability (11 x 9s files)
* Tiered Storage Available
* Lifecycle Management
* Versioning
* S3 allows to encrypt your data
* S3 allows you to secure data using Access Control Lists and Bucket Policies
* S3 Storage Tiers/Classes
  + **S3**: (Durable, immediately available, frequently accessed) 99.99% availability, 99.999999999% durability, stored redundantly across multiple devices in multiple facilities and is designed to sustain the loss of 2 facilities concurrently
  + **S3 - IA (infrequently Accessed)**: (Durable, immediately available, infrequently accessed) for data that is accessed less frequently, but requires rapid access when needed. Lower fee than S3, but you are charged a retrieval fee. You may lose a file because is just 99.99% rather than 99.999999999%
  + **Reduced Redundancy Storage**: designed to provide 99.99% durability and 99.99% availability of objects over a given year. For data that is easily reproducible such as thumb nails, etc
  + **Glacier**: Very cheap, but used for archival only. It takes 3 -5 hours to restore from glacier.
* S3 Charges
  + Storage
  + Requests
  + Data Transfer Pricing
* Buckets can trigger the following events:
  + Simple Notification Service (SNS)
  + Simple Queue Service (SQS)
  + Or to a lambda function
* You can use S3 to host a website, but it MUST be static pages
  + Create a bucket and select Properties
  + Enable website hosting and provide index and error document page
* All the documents and the bucket itself is private by default.
* To recognize the difference between a website on S3 and a normal bucket
  + https://s3-us-east-1.amazonaws.com/my-bucket -name: This is a bucket
  + <http://myownsite.s3-website-us-east-1.amazonaws.com>: This is a site

**Cross Origin Resource Sharing and Versioning**

* It can be enabled to allow referencing a file stored in a bucket from a web site
* Once Versioning a bucket is enabled, it cannot be disabled just Suspended
* It keeps copies of the files so the bucket size could increase rapidly
* Versioning Stores all versions of an object (including all writes and even if you delete an object)
* Objects can be backed up across different regions when versioning is enabled on the source bucket
* Versioning MFA Delete capability, which uses multi-factor authentication, can be used to provide an additional layer of security

**S3 Lifecycle**

* Is set on a bucket
* Can transition files to Infrequent Access Storage Class, but need to wait at least 30 days
* Can transition to Glacier next day if coming from standard, but 30 days if coming from Infrequent Access Storage Class
* Can be deleted
* Can be used in conjunction with versioning or by itself
* Can be applied to current versions and previous versions
* Following actions can now be done:
  + Transition to the Standard - Infrequent Access Storage Class (128 Kb and 30 days after the creation data)
  + Archive to the Glacier Storage Class (30 days after IA, if relevant)
  + Permanently Delete

**CloudFront**

* What is CDN? A Content Delivery Network is a system of distributed servers (network) that deliver web pages and other web content to a user based on the geographic location of the user, the origin of the webpage and a content delivery server
* CloudFront Key Terminology:
  + **Edge Location** - This is the location where content with be cached. This is separate to an AWS Region/Availability Zone
  + **Origin** - This is the origin of all the files that the CDN will distribute. This can be either an S3 Bucket, an EC2 instance, an Elastic Load Balancer or Route53
  + **Distribution** - This is the name given the CDN which consists of a collection of Edge Locations
  + **Web Distribution** - Typically used for websites
  + **RTMP** - Used for Media Streaming
* Edge locations are not just READ only, you can write to them too (ie put an object onto them)
* Objects are cached for the life of the TTL (Time to Live) in seconds, default is 24 hours
* You can clear cached object, but you will be charged
* Things to keep in mind with CDN:
  + When you setup your CDN you can protect access to your site by setting the Restrict Viewer Access (Use Signed URLs or Signed Cookies) to Yes
  + Once the CDN is created you can change the behavior. For instance to get all the jpg files from bucket A and all the text files from bucket B
  + You can create Custom error pages
  + You can restrict access to the page based on regions. You can set the white and black list by country. You cannot not set both you need to choose one or the other

**S3 Security and Encryption**

* By default, all new created buckets are PRIVATE
* You can set up access controls to your buckets using:
  + **Bucket Policies**: they are bucket wide
  + **Access Control Lists**: This drills down to objects
* S3 buckets can be configured to create access logs which log all requests made to the S3 bucket. This can be done to another bucket
* Encryption:
  + **In Transit**: Encrypts the data when is in transit. This is done using SSL/TLS
  + **At Rest**: The data is encrypted in the server, client, or both
    - *Server Side Encryption*: This is done using:
      * S3 Managed Keys - **SSE-S3**. This is all managed by AWS. Each object is encrypted using keys and the master key is also encrypted for extra protection.
      * AWS Key Management Service, Managed Keys - **SSE-KMS**. Same as above, plus separate permissions of the envelope key which is what protects the data. It also allows you to log who accessed the bucket
      * Server Side Encryption With Customer Provided Keys - **SSE-C**. You managed the encryption keys and amazon manages the encryption itself
    - *Client Side Encryption*: You encrypt the data yourself on the client side

**Storage Gateways**

* Storage Gateway Types:
  + **Gateway Stored Volumes**
    - Entire Dataset is stored on site and is asynchronously backed up to S3 (two sets of data one if the office and the other in AWS S3)
  + **Gateway Cached Volumes**
    - Entire dataset is stored on S3 and the most frequently accessed data is cached on site
  + **Gateway Virtual Tape Library (VTL)**
    - Used for backup and uses popular backup applications like NetBackup, Backup Exec, Veam, etc.

**Import/Export Disk**

* Import to EBS
* Import to S3
* Import to Glacier
* Export from S3 only

**Import/Export Snowball**

* Import to S3
* Export to S3

**S3 Transfer Acceleration**

* Utilises the CloudFront Edge Network to accelerate your uploads to S3. Instead of uploading directly to your S3 bucket, you can use a distinct URL to upload directly to an edge location which will then transfer that file to S3. You will get a distinct URL to upload to
* This transfer acceleration feature has a cost involved

**Database Essentials**

* Read all the RDS [FAQs](https://aws.amazon.com/rds/faqs/)
* RDS is an Online Transaction Processing (OLTP)
* Relational Database Types:
  + SQL Server
  + Oracle
  + MySQL Server
  + PostgreSQL
  + Aurora
  + MariaDB
* Non Relational Database (DynamoDB)
  + Database
    - Collection = Table
    - Document = Row
    - Key Value Pairs = Fields
* What is Data Warehousing?
  + Used for business Intelligence, tools like Cognos, Jaspersoft, SQL Server Reporting Services, Oracle Hyperion, SAP NetWeaver
  + Used to pull in very large and complex data sets. Usually used by management to do queries on data (such as current performance vs targets, etc)
* OLTP Vs OLAP
  + Online Transaction Processing (OLTP) differs from Online Analytics Processing (OLAP) in terms of the types of queries run
* OLTP Example: Get order 123; pulls up a row of data such as Name, Date, Address, etc
* OLAP Example
  + Pulls in large number of records
  + Performs multiple requests to the database
  + Think of Machine Learning
* To avoid impacting operating DB, normally OLAP works on a copy of the database
* What is **ElasticCache**?
  + Is a web service that makes it easy to deploy, operate, and scale an in-memory cache in the cloud. The service improves the performance of web applications by allowing you to retrieve information from fast, managed, in-memory caches, instead of relying entirely on slower disk-based databases.
* **ElasticeCache** supports two open-source in-memory caching engines:
  + Memcached
  + Redis
* What is **Database Migration Service (DMS)**?
  + Allows you to migrate your production database to AWS. once the migration has started, AWS manages all the complexities for the migration process like data type transformation, compression, and parallel transfer (for faster data transfer) while ensuring that data changes to the source database that occur during the migration process are automatically replicated to the target
  + AWS schema conversion tool automatically converts the source database schema and a majority of the custom code, including views, stored procedure, and functions, to a format compatible with the target database

**DynamoDB 101**

CRITICAL TO READ THE [FAQs](https://aws.amazon.com/dynamodb/faqs/) for DynamoDB (the most important aspect of the test)

* Is a NoSQL database service for all applications that need consistent, single-digit-millisecond latency at any scale. It is fully managed database and supports both document and key-value data models.
* Quick facts about DynamoDB
  + Stored on SSD storage so is fast
  + Spread across 3 geographically distinct data centers
  + Read Models:
    - **Eventual Consistent Reads** (default) (data updated within a second)
      * Consistency across all copiers of data is usually reached within a second. Repeating a read after a short time should return the updated data (Best read performance)
    - **Strongly Consistent Reads** (needs to be consistent all the time
      * A strongly consistent read returns a result that reflects all writes that received a successful response prior the read
* The Basics:
  + Tables
  + Items equivalent to a row of data in table
  + Attribute equivalent to columns or fields

{

“UniqueID”: 1975,

“FirstName”: “Allan”,

“Surname”: “Brown”,

“Age”: 41

}

{

“UniqueID”: 1976,

“FirstName”: “John”,

“Surname”: “Snow”,

“Age”: 46

}

* DynamoDB can store up to 35 levels of nesting objects
* Pricing:
  + Provisioned Throughput Capacity
    - Write Throughput $0.00665 per hour for every 10 units
    - Read Throughput $0.00665 per hour for every 50 units
  + First 25 GB stored per month is free
  + Storage after that costs $0.25 per GB per month

**DynamoDB Indexes and Streams**

* Primary Keys types:
  + **Single Attribute** (think unique ID)
    - Partition Key (Hash key) composed of one attribute
  + **Composite** (think unique ID and a date range)
    - Partition Key & Sort Key (Hash and Range) composed of two attributes
* DynamoDB uses the partition key’s value as input to an internal hash function. The output from the hash function determines the partition (this is simple the physical location in which the data is stored)
* No two items in a table can have the same partition key value unless a sort key is provided (Composite) and they need to be unique
* All items with the same partition key are stored together, in sorted order by sort key value
* Indexes
  + **Local Secondary Index**:
    - Has the SAME Partition key, different sort key
    - Can ONLY be created when creating a table. They cannot be removed or modified later
  + **Global Secondary Index**
    - Has DIFFERENT Partition key and different sort key
    - Can be created at table creation or added LATER
  + You can have up to 5 local secondary indexes per table and up to 5 global secondary indexes
* **DynamoDB Streams:** Used to capture any kind of modification of the DynamoDB Tables
  + If a new item is added to the table, the stream captures an image of the entire item, including all of its attributes
  + If an item is update, the stream captures the “before” and “after” image of any attributes that were modified in the item
  + If an item is deleted from the table, the stream captures an image of the entire item before it was deleted
  + DynamoDB Streams stores the data for only 24 hours
  + This is what triggers events and sends them to lambda for instance
  + Tables contents can be exported to a CSV files as well as individual entries
* Triggers can be used to replicate data, send notifications, execute a lambda function, etc
* What is a Query?
  + A Query operation finds items in a table **using only primary key** attribute values. You must provide a partition attribute name and a distinct value to search for
  + You can optionally provide a sort key attribute name and value, and use a comparison operator to refine the search results
  + By default, a Query returns all of the data attributes for items with the specific primary key(s); however you can use the ProjectionExpression parameter so that the Query only returns some of the attributes, rather than all of them
  + Query results are always sorted by the sort key in ascending order. To reverse the order need to set the ScanIndexForward parameter to false.
  + By default is eventually consistent but can be changed to strongly consistent
* What is a Scan?
  + A Scan operation **examines every item in the table**. By default, a Scan returns all of the data attributes for every item; however, you can use the ProjectionExpression parameter so that the Scan only returns some of the attributes, rather than all of them
* Generally a Query operation is more efficient that a Scan operation
* A Scan operation always scans the entire table, then filters out values to provide the desired result, essentially adding the extra step of removing data from the result set
* For quicker response times, design your tables in a way that you can use only Query, Get, or BatchGetItem APIs instead

**DynamoDB Provisioned Throughput Calculations**

* Unit of Read provisioned throughput
  + All reads are rounded up to increments of 4 KB
  + Eventually Consistent Reads (default) consists of 2 reads per second
  + Strongly Consistent Reads consists of 1 read per second
* Unit of Write provisioned throughput
  + All writes are 1 KB
  + All writes consists of 1 write per second
* The magic formula:
  + Throughput =
    - **(Size of Read rounded to nearest 4KB chunk / 4KB) X num items**
  + Divide by 2 if eventually consistent
  + **Question**: You have an application that requires to read 10 items of 1 KB per second using eventual consistency. What should you set the read throughput to?
    - First we calculate how many read units per item we need
    - 1 KB rounded to the nearest 4 KB increment = 4
    - 4 KB / 4KB = 1 read unit per item
    - 1 x 10 read items = 10
    - Using eventual consistency we get 10 / 2 = 5
    - 5 units of read throughput
  + **Question**: You have an application that requires to read 10 items of 6KB per second using eventual consistency. What should you set the read throughput to?
    - How many reads ceiling(6 KB / 4 KB) = 2
    - 2 X 10 = 20 reads
    - Using eventually consistency 20 / 2 = 10
    - 10 units of read throughput
  + **Question**: You have an application that requires to read 5 items of 10 KB per second using eventual consistency. What should you set the read throughput to?
    - Nearest increment of 4 is 12
    - 12 / 4KB = 3
    - 3 x 5 reads = 15
    - Is eventual consistency so 15 / 2 = 7.5 or rounded to 8
    - 8 read throughput
  + **Question**: You have an application that requires to read 5 items of 10 KB per second using strong consistency. What should you set the read throughput to?
    - Nearest increment = 12
    - 12 / 4 = 3
    - 3 \* 5 = 15
    - 15 read throughput
  + **Question**: You have an application that requires to write 5 items, with each item being 10 KB in size per second. What should you set the write throughput to?
    - Need to write 5 x 10 KB = 50 KB per second
    - Write throughput = 50 units
  + **Question**: You have an application that requires to write 12 items, with each item being 100 KB in size per second. What should you set the write throughput to?
    - 12 x 100 KB = 1200 KB per second
    - Write throughput is 1200 units
* If the throughput exceeds the amount set to you get a “400 HTTP Status Code: **ProvisionedThroughputExceededException**” You exceeded your maximum allowed provisioned throughput for a table or for one or more global secondary indexes

**DynamoDB and Web Identity Providers**

* First you authenticate with the ID provider (facebook)
* Then you will receive an identity token once you provide the appropriate username and password
* Using the AssumeRoleWithWebIdentity request API and provides the providers token and the ARN for the IAM Role and submit it to AWS Security Token Service
* The response is a temporary security credentials that by default lasts one hour
* The temporary security credentials contain the following:
  + AcessKeyID, SecretAccessKey, SessionToken
  + Expiration (time limit default 1 hour)
  + AssumeRoleID
  + SubjectFromWebIdentityToken
* Now you can access DynamoDB from between 15 minutes to 1 hour

**Other Important Aspects of DynamoDB**

* **Conditional Writes** (i.e. if item = $20 then update to $22)
  + Conditional writes are idempotent. This means that you can send the same conditional write request multiple times, but it will have no further effect on the item after the first time DynamoDB performs the specified update. For example, suppose you issue a request to update the price of a book by 10%, with the expectation that the price is currently $20. However, before you get a response, a network error occurs and you don’t know whether your request was successful or not. Because a conditional update is idempotent operation, you can send the same request again and DynamoDB will update the price only if the current price is still $20
* **Atomic Counters**
  + DynamoDB supports atomic counters, where you use the UpdateItem operation to increment or decrement the value of an existing attribute without interfering with other write requests. (All write requests are applied in the order in which they were received). For example, a web application might want to maintain a counter per visitor to their site. In this case, the application would need to increment this counter regardless of its current value
  + Atomic counters are NOT idempotent
* **Batch Operations**
  + If our application needs to read multiple items, you can use the BatchGetItem API. A single BatchGetItem request can retrieve up to 1 MB of data, which can contain as many as 100 items. In addition, a single BatchGetItem request can retrieve items from multiple tables.

**Simple Queue Service (SQS)**

* SQS is a web service that gives you access to a message queue that can be used to store messages while waiting for a computer to process them
* SQS messages can be delivered multiple times and in any order
* The default visibility time is 30 seconds but can be set to up to 12 hours
* The maximum retention period of SQS messages is 14 days
* When you receive a message from the queue and begin processing it, you may find the visibility timeout for the queue is insufficient to fully process and delete that message. To give yourself more time to process the message, you can extend its visibility timeout by using the ChangeMessageVisibility action to specify a new timeout value, Amazon SQS restarts the timeout period using the new value
* SQS Long Polling is a way to retrieve message from your SQS queues. While the traditional SQS Short polling returns immediately, even if the queue being polled is empty, SWS Long Polling doesn’t return a response until a message arrives in the queue, or the long poll times out
* Maximum Long Poll Timeout = 20 seconds
* Long Polling saves money as it does not keep sending requests asking for messages to the server
* You can decouple applications using SQS
* Messages can contain up to **256 KB** of text in any format, but the billing is based on 64 KB chunks
* First 1 million Amazon SQS requests per month are free
* $0.50 per 1 million Amazon SQS requests per month thereafter
* A single request can have from 1 to 10 messages, up to a maximum total payload of 256 KB
* SQS does not guarantee first in, first out messages. If order is required then the producer needs to place sequencing information in each message
* All the **messages needs to be pulled** from the Queue, they are never pushed out
* Example of a decoding image files queue:
  1. Asynchronously pulls the task messages from the queue
  2. Retrieves the named file
  3. Processes the conversion
  4. Writes the image back to Amazon S3
  5. Writes a “task complete” message to another queue
  6. Deletes the original task message
  7. Checks for more messages in the worker queue
* SQS can use auto scaling services
* To fan out messages using SQS and SNS. A message from SQS gets inserted into SNS and multiple instances are monitoring that Topic then all of them would be notified

**Simple Notification Service (SNS)**

* SNS is a web service that follows the publish/subscribe messaging paradigm, with notifications being delivered to clients using a push mechanism that eliminates the need to periodically check or poll for new information and updates. With simple APIs requiring minimal up-front development effort, no maintenance or management overhead and pay-as-you-go pricing, Amazon SNS gives developers an easy mechanism to incorporate a powerful notification system with their applications
* SNS can push notifications to Apple, Google, Fire OS, and Windows devices, as well as Android devices in China with Baidu Cloud Push
* It can also deliver notifications by SMS text message or email to Amazon SES or to any HTTP endpoint
* To prevent messages from being lost all messages published to Amazon SNS are stored redundantly across multiple Availability Zones
* SNS uses Topics as an access point for allowing recipients to dynamically subscribe
* SNS Benefits:
  + Instantaneous, push-based delivery (no polling)
  + Simple APIs and easy integration with applications
  + Flexible message delivery over multiple transport protocols
  + Inexpensive, pay-as-you-go model with no up-front costs
  + Web-based AWS Management Console offers the simplicity of a point-and-click interface
* SNS Pricing:
  + Users pay $0.5 per 1 million Amazon SNS requests
  + $0.06 per 100,000 Notification deliveries over HTTP
  + $0.75 per 100,000 Notification deliveries over SMS
  + $2.00 per 100,000 Notification deliveries over Email
* The data format used is JSON
* The message will always contain the following attributes:
  + Type,
  + MessageId,
  + TopicArn (Amazon Resource Name),
  + Subject,
  + Message,
  + Timestamp,
  + SignatureVersion,
  + Signature,
  + SigningCertURL,
  + UnsubscribeURL,
  + MessageAttributes
* SNS Protocls Include:
  + HTTP
  + HTTPS
  + Email
  + Email-JSON
  + Amazon SQS
  + Application
* Messages can be customized for each protocol

**Amazon Simple Workflow Service (SWF)**

* Is a web service that makes it easy to coordinate work across distributed application components. It enables applications for a range of use cases including media processing, web application backends, business process workflows, and analytics pipelines, to be designed as a coordination of tasks. Tasks represent invocations of various processing steps in an application which can be performed by executable code, web service calls, human actions, and scripts
* SWF Workers are programs that interact with Amazon SWF to get tasks, process received tasks, and return the results
* SWF Decider is a program that controls the coordination of tasks, i.e. their ordering, concurrency, and scheduling according to the application logic
* SWF ensures that a task is assigned only once and is never duplicated
* SWF maintains the application’s state durably, workers and deciders don’t have to keep track of execution state. They can run independently and scale quickly
* SWF Domains: Your workflow and activity types and the workflow execution itself are all scoped to a domain. Domains isolate a set of types, executions, and task lists from others within the same account. You can register a domain by using the AWS Management Console or by using the RegisterDomain action in the Amazon SWF API
* Maximum Workflow can be 1 year and the value is always measured in seconds
* SWS Vs SQS
  + SWF presents a task-oriented API, whereas SQS offers a message-oriented API
  + SWF ensures that a task is assigned only once and is never duplicated. SQS, you need to handle duplicated messages and may also need to ensure that a message is processed only once
  + SWF keeps track of all the tasks and events in an applications. With SQS you need to implement your own application-level tracking especially if your applications uses multiple queues

**Amazon Cloud Formation(ACF)**

* Cloud Formation itself is free, but you will get charged for the resources it creates and use
* To identify the ACF need to look the JSON template/scripts
* To get outputs you can use the “**Fn::GetAtt**” function
* If there is an error by defaults ACF will roll-back and delete all the security group as well as any other resource it created

**Amazon Elastic Beanstalk**

* ElasticBean comes preconfigured with
  + IIS
  + Node.js
  + PHP
  + Python
  + Ruby
  + Tomcat
  + Docker
    - GLassFish
    - Python
  + Generic Docker
* You don’t pay for Elastic Beanstalk, but you pay for the resource it uses

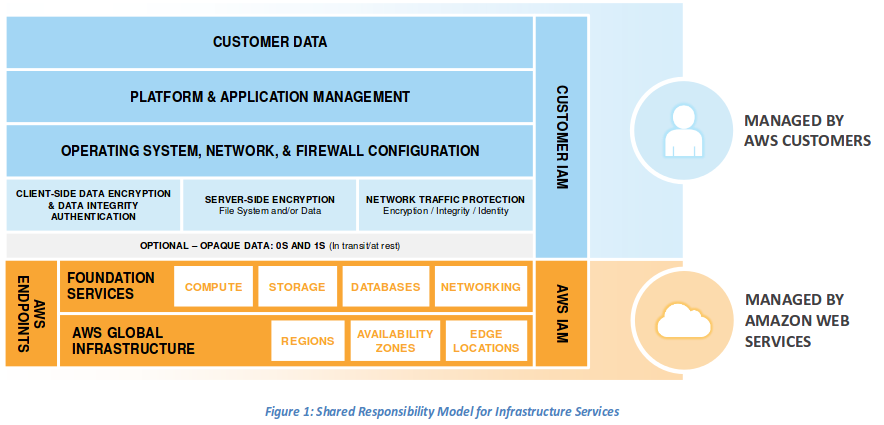
**Virtual Private Cloud (VPC)**

* Amazon Virtual Private Cloud lets you provision a logically isolated section of the Amazon Web Service Cloud where you can launch AWS resources in a virtual network that you define. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways
* You can easily customize the network configuration for your Amazon VPC. For example, you can create a public-facing subnet for your web servers that has access to the Internet, and place your backend systems such as databases or application servers in a private-facing subnet with no Internet access. You can leverage multiple layers of security, including security groups and network access control lists, to help control access to Amazon EC2 instances in each subnet
* Just because you have an instance in a public subnet does not mean it has access to the Internet. The instance need either have an Elastic IP Address or have an Elastic Load Balancer
* You are allowed up to 5 VPCs in each AWS region by default
* What can you do with a VPC?
  + Launch instances into a subnet of your choosing
  + Assign custom IP Address ranges in each subnet
  + Configure route tables between subnets
  + Create internet gateways and attach them to subnets (or not)
  + Much better security control over your AWS resources
  + Instance security groups
  + Subnet network access control lists (ACLS)
* Default VPC vs Custom VPC
  + Default VPC is user friendly allowing you to immediately deploy instances
  + All subnets in default VPS can an internet gateway attached
  + Each EC2 instance has both a public and a private IP address
  + If you delete the default VPC the only way to get it back is to contact AWS
* VPC Peering
  + Allows you to connect one VPC with another via a direct network route using private IP addresses
  + Instances behave as if they were on the same private network
  + You can peer VPC’s with other AWS accounts as well as with other VPCs in the same account
  + Peering is in a star configuration, ie 1 central VPC peers with 4 others. NO TRANSITIVE PEERING!!!
* **CIDR**: Cloud Inter-Domain Routing which is the IP Address block assignment
* If you set the Tenancy to Dedicated when creating a new VPC, all the EC2 instances will be dedicated rather than a shared hardware resource
* Route tables are created automatically when you create a VPC
* Subnets can be in **one and only one** Availability Zone.
* The default routing table allows to communicate between subnets (Destination 10.0.0.0/10 ⇒ Target local)
* Internet Gateways allows Internet access to the EC2 Instances
  + They are detached by default and needs to be attached to a VPC after creation
  + There can be **only one Internet Gateway per VPC**
* To allow all EC2 instances to have access to the Internet we need to create a new Route in the Route tables
  + The target needs to be the VPC Internet gateway
  + The Destination needs to be 0.0.0.0/0 which means all traffic
* You normally associate one Subnet with the Route
* Security Groups can exists across different Availability Zones while Subnets cannot
* If EC2 has a private IP only it means it is not Internet accessible
* **How to create a VPC**
  + Create VPC give it a name and set the CIDR block to 10.0.0.0/16 leave tenancy to default so we do not use dedicated hardware
  + A Route Table is created automatically allowing any EC2 to communicate from any of the Subnets to any other EC2 in any Subnet
  + Create Subnets, one for each availability zone as in:
    - Set the name tag to 10.0.1.0 - AZ
    - Select the newly created VPC
    - Set the AZ to the appropriate value
    - Set the CIDR block to 10.0.**n**.0/24 where n is associated to that subnet
  + Create a new Internet Gateway to allow access to the Internet and attach it to the VPC. There can be only ONE Internet Gateway per VPC
  + Create a new Route Table and set the VPC to the newly created VPC
  + Add a Route to the Route Table with the following settings:
    - Destination: 0.0.0.0/0 which means full access
    - Target: The created Internet Gateway from previous steps
  + Associate the Route Table to a Subnet (Subnet Associations tab) to allow it to have access to the Internet
  + You can only associate one Route Table to a Subnet
* **Network Access Control List (NACL)**
  + NACL overwrites Security Group. For instance, a Security Group allows access to port 80, but the NACL does not then none of the EC2 instances in that Subnet will be able to access port 80
  + If no NACL is set, then the default one is assigned to the Subnet
  + Rules with lower rule numbers are evaluated before rules with higher numbers
  + It is recommended to use rule number multiple of 100 in order to add more rules if needed
  + By default all Inbound and Outbound Settings are denied
  + Subnet can only have one Network Access Control List, but A Network Control Access List can have multiple Subnets
* **Network Address Translation (NAT)**
  + NAT translates from one subnet address to another. This is useful to allow some instances to have access to the Internet for example
    - Create a new Security Group
    - Set the Inbound to be from the desired subnet (10.0.2.0/24) for each protocol (HTTP, HTTPS, etc)
    - Set the Outbound to be any destination (0.0.0.0/0) for each protocol (HTTP, HTTPS, etc)
    - Create a new NAT instance by selecting AMI Family rather than regular way
  + If NAT is not performing properly one way to improve its performance is by creating a NAT on a better instance type
  + Typically all EC2 instances perform a check to make sure the traffic is coming from the node itself (its own IP address). In the case of the NAT we need to disable the check because the traffic can come and go from different nodes or instances. This is disable in the ***Actions - Networking - Change Source/Dest Check***
  + To enable the private subnet to speak to the NAT we create a new Route.
    - Select the default Route Table for the VPC
    - Select the Routes tab and Edit and add another route
    - Set the Destination to all traffic (0.0.0.0/0) and the NAT VM as the Target

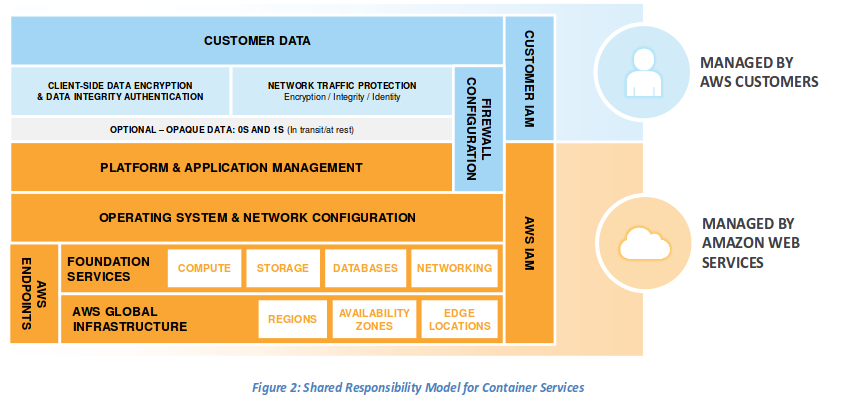
**Shared Responsibility Model**

Take a look at the [AWS Security Best Practices](http://media.amazonwebservices.com/AWS_Security_Best_Practices.pdf) paper

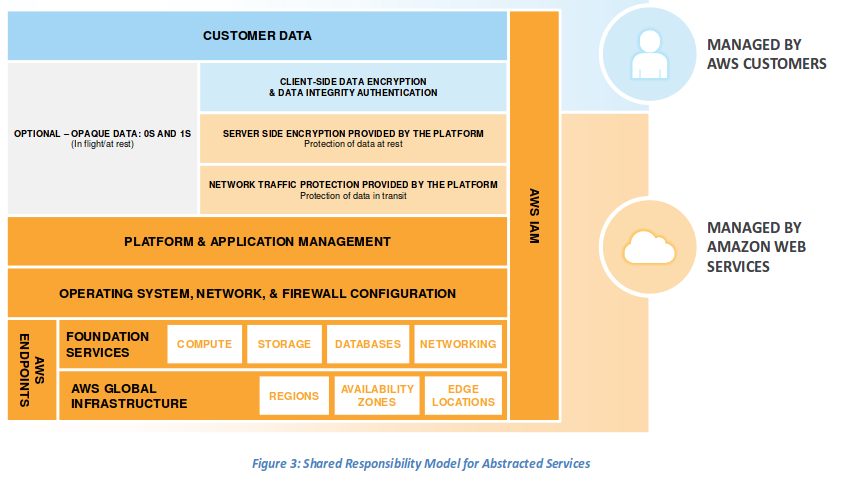
Responsibility Model for infrastructure services such as EC2, EBS, and VPC



Shared Responsibility Model for Container Services such as EMR, RDS



Shared Responsibility Model for Abstracted Service such as S3, DynamoDB, and possibly Lambda



**Exam Practicalities**

* 80 minutes in length
* 55 question in the exam
* Conducted online at an approved centre
* Register online at <https://www.webassessor.com>
* Need to be there 15 minutes early and provide the proctor with the Test Taker Authorization Code
* Bring 2 forms of identification (Government ID + credit card or something)