I've passed AWS Certified Big Data Specialty, after 4 months preparation! This certification exam is more difficult exams I've faced, since I didn't have recommended 5 years of experience or 2 years of AWS experience to pursue certification. I want to share with you my studying strategy and notes to help you pass the exam. You have 3 hours to answer 65 scenario based questions.

Refer to the exam blueprint and sample questions.

Most of the questions are from Kinesis, Dynamo db, Redshift, EMR Services. Other services like IOT, Glue, Athena will appear on exam with one or two points each, so they are worth reading. Security plays major part in exam.

Courses: 1)A Cloud Guru's AWS Big Data Certification course (Complete Course). 2)Udemy's AWS Certified Big Data Specialty 2019 (Complete Couse). 3)Linux Academy AWS Big Data – Specialty Certification (Used free trail)

- I've gone through all videos thrice and made notes for every important point and keyword. Practice each and every Service with different combinations to possible extent. It'll cost around $10 to practice (including using Free tier), but it makes understand the services better which makes exam easier.
- I've gone through reinvent latest videos for most of the service. This was super helpful, I've created playlist, If you're interested you can watch them at YouTube and search for other reinvent videos as well. I've read 7 Aws whitepapers thrice. It provides brief of each service in application.
  1) Overview Big Data Analytics Options on AWS , Blog
  2) EMR – Best Practices of Amazon EMR
  3) Redshift – Enterprise Data Warehousing on AWS
  4) Kinesis - Streaming Data Solutions on AWS with Amazon Kinesis
  5) Dynamo Db - Comparing the Use of Amazon Dynamo DB and Apache HBase for NoSQL
  6) Dynamo Db - Best Practices for Migrating from RDBMS to Amazon Dynamo DB
- I've read all the blog posts mentioned in course. Please read them for practical examples.
- Security is major part of exam. It was difficult for me to handle security, because I lack experience. I've practiced security in service in console and made notes for each of them. This made to answer security questions without much difficulty. I've scored 84% in security domain.

Dynamo Db

- Fully managed No Sql DB
- Partition key or partition key + sort key
- Capacity is measured in WCU and RSU
- Write:1 write per second per item of 1 KB.
- Read: Strong consistent reads 1 read/sec per item of 4 kb eventually consistent reads 2 reads/sec per item of 4 kb.
- Dynamo db saves data in partitions. Dynamo db support only 3000 RCU/1000WCU or 10 GB for each partition.
- Creates new partition automatically based on limits but it **is not decreased automatically**. This will have performance impact.
- No of partitions = **max ((required RCU's/3000 = Required WCU's/1000), data in GB/10)**

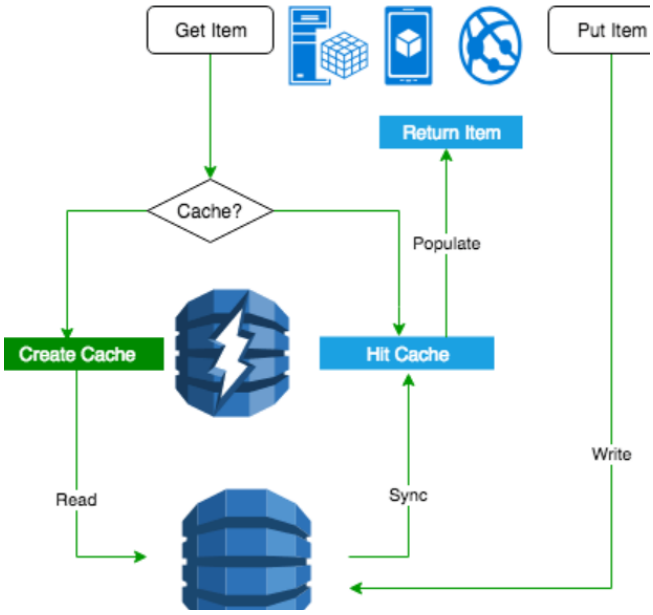
Partition key: It is important to distribute data evenly across partitions which increases performance. Recommendations for Partition Key:

- Select **high cardinality** values such as customerid, emailid etc.
- Use **composite** attributes like customerid+productid+countrycode as partition key.
- Use **cache** for high volume of reads.
- **Don't use database generated** values like identities, transaction id's. Instead use **conditional writes.**
- Use **write shading technique** (adding random and calculated values) to distribute data evenly across partitions to handle heavy read and write transactions and distribute data evenly.

Sort Key: Sort key is included to improve the query access patterns and performance.

- **Use low cardinality** values as sort key like order date.
- Sort keys can be implemented to capture version history and retrieve **latest version** efficiently.

DAX: Dynamo db accelerator is caching service which enables to perform **heavy read** loads with fast response times with microsecond latency. DAX support only eventually consistent reads.



- DAX save cost of read intensive applications by **instead of increasing RCU.** We can have multiple DAX cluster to same dynamo db table.

Burst Capacity: Dynamo db provides a flexibility in partition throughput provision by providing burst capacity. Whenever dynamo db is not using partitions throughput, it reserves a portion of that **unused capacity** for later burst of throughput to handle usage spikes

Adaptive Capacity: Adaptive capacity is feature that allows to run **imbalanced workloads indefinitely**. It minimizing throttling due to throughput exceptions and also reduces to cost by enabling you to provide only capacity you need.

Indexes: Two types

1)GSI: Global Secondary Indexes. GSI's can have **different partition** key or partition + sort key than main table. GSI's can be created at **any time**. GSI's have **different** RCU and WCU. GSI queries supports only eventually consistent reads.

Using GSI sparse Index: add a Boolean attribute or sparse index. It is useful for queries over a small subsection of table.

Ex: Top 10 or number of photos.

GSI overloading: Adding different fields to index to cater different queries.

GSI Sharding: To enable selective queries across the entire key space, you can use write Sharding by adding attribute containing (0-N) value to every item that you will GSI partition key.

2)LSI: Local Secondary Indexes. LSI's should be created at **time of table creation**. LSI's have **same partition** key but different sort key. LSI's can support strong consistent Reads. Only 5 LSI are possible.

Global tables: Global table is a collection of one or more replicas. Any given global tables can only have one replica for one AWS region. All global tables should be of identical name and Write capacity should be identical

TTL: Time to Live is feature supported by dynamo db which runs a **background** job to delete the records based **epoch** time value. This will not affect any WCU/RCU.

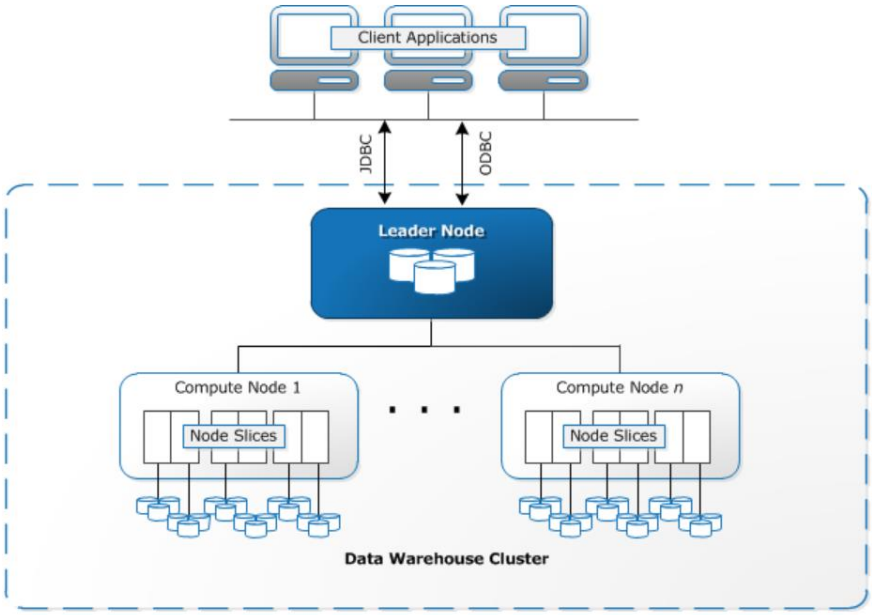
Dynamo Db Streams: Ordered records of updated to dynamo dB table. When dynamo db streams are enabled, it records changes to a table and stores those values for **24 hrs.** Streams can enabled by API or console, but read or processed via stream endpoint and API requests.

There are 4 stream views:

1. KEYS_ONLY — Only the key attributes of the modified item.
2. NEW_IMAGE — The entire item, as it appears after it was modified.
3. OLD_IMAGE — The entire item, as it appeared before it was modified.
4. NEW_AND_OLD_IMAGES — Both the new and the old images of the item.

Redshift

- Massively Parallel Processing, Columnar Storage Data warehouse service.
- Contains Leader nodes and compute nodes. Each compute node has **node Slices**, a logical CPU
- Distribution Styles and Sort keys plays key role in Performance.



Sort Key: Redshift stores data in sorted format based select sorted column.

Types:

1)Single Sort Key: mention of the required column which involve in your queries.
Example: If you query on most recent data frequently, use timestamp as sort key. If you have frequent range filtering or equality, use that column as Sort key

2)Compound Sort Key: one or more columns are included in the sort key. It is most useful when a query filter applies conditions as filters or joins. Performance decreases when query depend only on secondary sort key column. Compound sort key will speed up the group by, order by, partition by operations.

3)Interleaved Sort Key: It gives equal importance to all the columns in sort key. This type of sort key gives good performance when we use multiple queries that have different columns for filter. Interleaved Sort key is more effective with large tables.

Constraints: We can define constraints like Primary key, Foreign Key, unique, not null etc., even though redshift won't enforce them, it helps in creating good query plan and increases performance.

Copy command: This is most common way to load the data in redshift. Copy command loads data in parallel from S3, EMR, Dynamo Db or any other database or from remote host.

- Don't use multiple copy commands to load to load one table. Copy command loads data in parallel from multiple files. Number of files = equal to number of slices or multiple of (number of slices).
- File sizes should be between 1MB – 1 GB. For optimum use 1mb -125 mb.
- When copy command is not an option, use multi insert than single row inserts or use bulk insert file like insert into or create table as statements.

- Load data in sorted order to avoid vacuum medium. If you need to large data, load it in sequential order to eliminate need of vacuum.
- COPY command supports Server Side encryption with S3 manages Key, Server Side Encryption – KMS, Client Side encryption using Client side asymmetric master key and Copy command will not support Server Side encryption using customer provided key, Client side with KMS managed customer master key and Client side with customer provided asymmetric master key.

**** Copy and sync data between Redshift and PostgreSQL through DBLink**

Unload: Unload command is used to unload data from Redshift. UNLOAD encrypted command automatically encrypts while unloading data from Redshift to S3. Unload Command supports Server Side Encryption – KMS, Client Side encryption using Client side asymmetric master key.

System tables:

- STL: these tables are logs. Redshift maintains connection logs, user logs and user activity logs.
- STV: Virtual tables that contain snapshot of current data and system catalogue which store metadata.
- STL_LOAD_ERROR and ETL_LOAD_ERROR_DETAILS tables are used to get error information while loading data with copy command.

Query Best Practices:

- Use **predicate** to restrict data.
- Avoid using functions in query predicates.
- Use **where** clause to restrict data.
- Use sort key in group by operations to aggregation. For complex aggregations use **Case Statement.**

Redshift Work Load Management(WLM): WLM is used to configure user groups and query groups.

- We can configure up to 8 queries and set the number of queries that run in each of those ques concurrently.
- Set up rules to route queries to particular ques based on user running query or labels.
- **Short Query Acceleration** prioritizes short queries, which enables short running queries run ahead of long running queries in dedicated space. If we **enable SCA**, we can reduce or eliminate WLM ques that are dedicated to run short queries.

Vacuum: Vacuum operations needs to be performed to resort the data after multiple or bulk insert. It helps to remove **deleted row space**. Vacuum operation is I/O intensive.

There are four types of Vacuum operations

1. Vacuum Sort: Sorts the specified table (or all tables in the current database) **without reclaiming space freed by deleted rows**. This option is useful when reclaiming disk space isn't important but re-sorting new rows is important. A SORT ONLY vacuum reduces the elapsed time for vacuum operations when the unsorted region doesn't contain a large number of deleted rows and doesn't span the entire sorted region.

2. Vacuum DELETE ONLY: Amazon Redshift automatically performs a DELETE ONLY vacuum in the background, so you rarely, if ever, need to run a DELETE ONLY vacuum. A VACUUM DELETE **reclaims disk space occupied by rows that were marked for deletion** by previous UPDATE and DELETE operations, and compacts the table to free up the consumed space. A DELETE ONLY vacuum operation doesn't sort table data.

3. Vacuum Reindex: Analysis the distribution of the values in **interleaved sort key** columns, then performs a full VACUUM operation.

4. Vacuum Full: Sorts the specified table (or all tables in the current database) and reclaims disk space occupied by rows that were marked for deletion by previous UPDATE and DELETE operations. VACUUM FULL is the default.

Deep Copy: Vacuum operation is I/O intensive and time consuming process if the table is huge. Instead of Vacuum operation, perform Deep Copy if the table is huge. Deep copy can perform by using table original DDL, Create table as and insert into commands.

Backup and recovery: Can be automatic or manual. Default snapshot retention period is 1 day and can be extended up to 35 days. Automatic snapshots will capture for 8 hrs or 5GB data change.

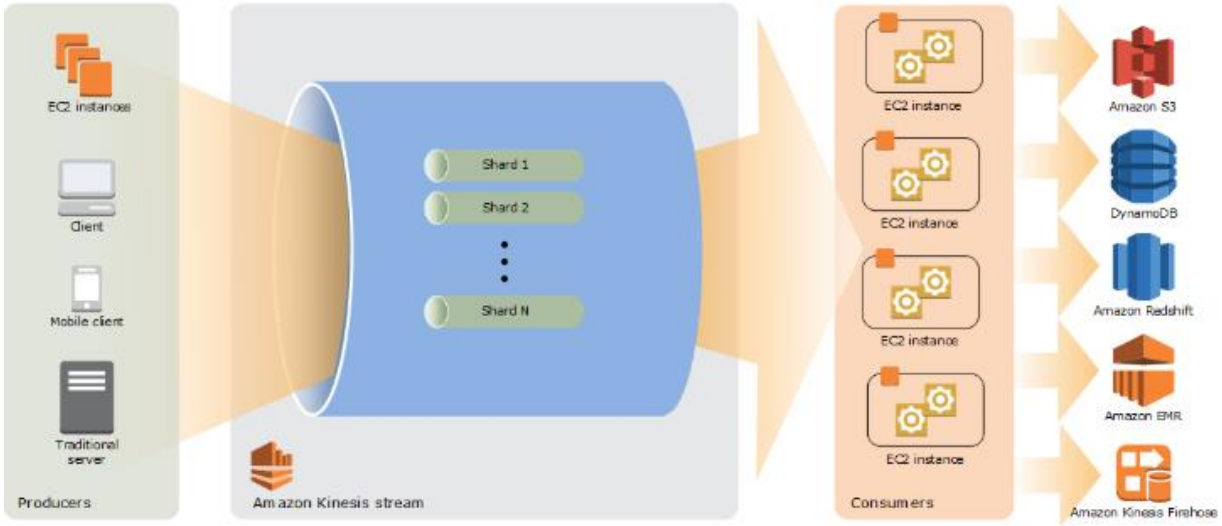
Redshift Spectrum: Amazon Redshift Spectrum is a feature within Amazon Web Services' Redshift data warehousing service that lets a data analyst conduct fast, complex analysis on objects stored on the AWS cloud. Can query data in its original format directly from S3 in same region.

- It supports AVRO, Parquet, text files, sequence file, Rc file, regex, serder, ORC, OpenCSV, Grok, Json etc. It supports gzip, snappy bzip compressions.
- Redshift external schema references external db and external data CatLog. We can external tables or databases in Athena, Glue CatLog or Hive metastore. By default, Redshift Spectrum stores external tables and external databases in Athena. Redshift spectrum should have permission to Athena, Hive/EMR, S3 and Glue.

Amazon Kinesis

Kinesis Data Streams

Input – SDK, KPL, Kinesis Agent, Kafka, Spark
Output – Spark, Kinesis Firehose, Lambda, KCL, SDK



Three Components:

1)Data Streams: Data stream is set of shards. Each shard has sequence of data records, shard supports 5 reads/sec per 2mb, 1000 write records of 1mb per sec.

- Use record is blob of data like UI event in Json. Kinesis data record consists of sequence no, partition Key and data blob. Partition key is used to group data within a stream. Sequence number is unique within stream assigned by kinesis data streams.
- Reshard: We can split/merge shards according to our requirements. Shard split increases the capacity of stream and increases cost as well. Generally, hot shards will be split. Shard merge reduces the unused capacity and decreases cost as well. Generally, cold shards will be merged.

2)Producers: Applications which produce data. It can be built in 3 ways.

a) Kinesis Agent: Kinesis is a standalone java software application that easily collects data and send data to stream. It is **simplest way to collect logs**.

- It can perform data collection, pre-processing and **light weight transformations** like CSV to JSON, Apache Log to JSON, Syslog to JSON, multiline – single line before writing data into streams.
- Kinesis agent handles file rotation, check pointing and retry upon failures. It can perform buffer based on time or size or number of records.
- We can configure agent to monitor multiple file directories and send data to multiple streams.

- Kinesis Agent publishes **custom cloudwatch** metrics with namespace of kinesis stream.

b) API: We can use Kinesis data streams with API with SDK for java. It includes methods like PutRecord, PutRecords to insert data into stream. We can monitor API using Cloud Trial.

For most cases we need to prefer KPL over API. Applications that cannot tolerate additional delay, have to use SDK.

C)KPL: KPL simplifies producer application development with high throughput

KPL performs

- Write to one or more kinesis data streams with automatic and configurable retry mechanisms.
- Collects records and uses putrecords method to write records to multiple shards per request.
- Aggregates use records to increase payload and throughput.
- Integrate with KCL to de-aggregate.
- It can be implemented in synchronous or asynchronous, asynchronous will be high performant.

3) Consumer: We have different consumers like

a) Lambda: we can use lambda functions to process the records in the stream.

b) Kinesis Analytics: We can use Analytics to process and analyses data in kinesis stream using SQL or java to enrich data or find anomalies, then we can write analysis results into another kinesis stream or firehose or lambda.

C) Firehose: Firehose can read and process kinesis data streams and send data to Redshift, S3, Elastic Cache and Splunk.

d) KCL: KCL is java library which consumes and process data. We can write custom applications with KCL library.

KCL performs

- Connects to streams
- Enumerates shards
- Co-ordinate with shard workers
- Instantiates a record processor for every shard.
- Pull and push records to processor
- Coordinates with KPL to de aggregate the records
- Uses dynamo db for check pointing the processed records. (So maintain enough WCU and RCU for dynamo db table to avoid throughput exceptions).
- It emits per KCL Application metrics, per worker metrics, per shard metrics.

** **KPL may throw ProvisionThoughputExceeded exceptions if the polling interval is too short. We can monitor KCL using Cloud watch.**

Kinesis data steams provide capabilities to use future objects to validate user records.

KCL Enhanced Fanout: Consumers that use enhanced fan-out in Amazon Kinesis Data Streams can receive records from a data stream with dedicated throughput of up to 2 MB of data per second per shard. It can be built from API or KCL. It provides 2MB/sec per shard per consumer. KCL automatically subscribe to consumer to all shards in the stream.

Metrics can be configured Kinesis Data streams to collect and push to cloud watch automatically every minute. We can capture basic stream level/shard level for every minute for free.

Kinesis Firehose:

Kinesis Firehose is simple way send data to **S3, Redshift, Elastic Cache and Splunk only** with no administration. It receives data from Kinesis data streams, kinesis Agent, Kinesis API, SDK, Cloudwatch Metrics, IOT.

- Firehose can buffer payload with size or time. For S3 ideal buffer size is 1MB-128 MB, for Elastic Cache 1Mb – 100 Mb and time can be buffered between 60- 900 seconds. Firehose will send data to S3 and executes copy command on data to move from S3 to Redshift.
- Firehose can covert input from Json to ORC or Parquet. It can perform light weight transformations using Lambda.
- Firehose can send raw to one bucket, transformed data to other bucket and transformation failed records to processing-error bucket.

- Firehose can also compress the data. It supports only Gzip, snappy and Zip compression.
- Firehose data retention period is maximum of 24 hrs.

Kinesis Analytics:

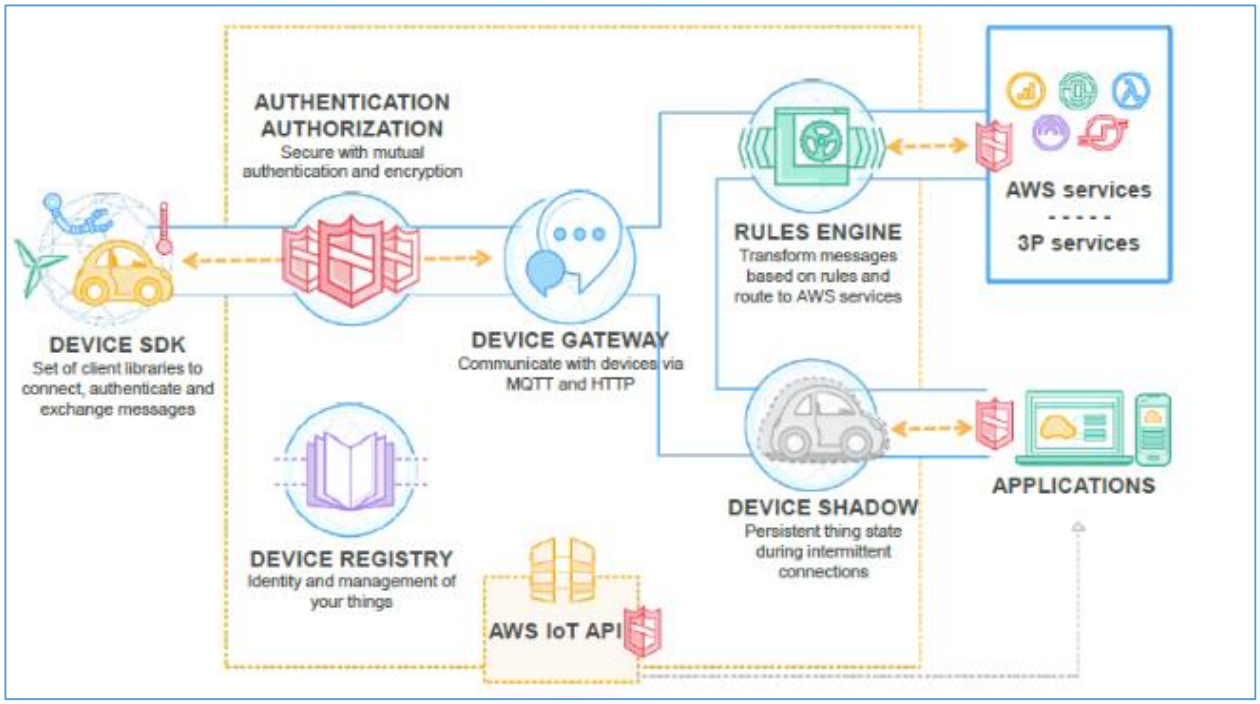
Input: Kinesis Data streams or Firehose only.
Output: Kinesis Data Streams or Firehose, other destinations via lambda. We can also use Apache Flink.

- We can use java to process and perform analytics and feed real time dashboards.
- It contains application code which is series of SQL statements, we can also use join data with reference tables to enrich the data.
- We can create in-application streams as needed to store intermediate query result.
- We can use continuous queries. A query over a stream executes continuously over streaming data. This continuous execution enables scenarios, such as the ability for applications to continuously query a stream and generate alerts or detect anomalies.
- SQL queries in your application code execute continuously over in-application streams. An in-application stream represents unbounded data that flows continuously through your application. Therefore, to get result sets from this continuously updating input, you often bound queries using a window defined in terms of time or rows. These are also called windowed SQL. It supports
  1)Stagger window: Using stagger windows is a windowing method that is suited for analyzing groups of data that arrive at inconsistent times. It is well suited for any time-series analytics use case, such as a set of related sales or log record or VPC flow logs.
  2)Tumbling window: When a windowed query processes each window in a non-overlapping manner, the window is referred to as a tumbling window. In this case, each record on an in-application stream belongs to a specific window. It is processed only once
  3)Sliding Window: we can define a distinct time-based or row-based window.

Error Handling: Reports runtime errors using in-application error streams called error streams.

Internet of Things(IOT)

IOT core contains components like Device gateway, message broker, Thing registry, Authentication and authorization, device shadow, Rules Engine.



Device Gateway: Servers as entry point for IOT devices connecting to AWS. Device Gateway allows devices to communicate effectively and securely. It supports MQTT, web sockets, HTTP.

Message Broker: Publishes/ Subscriber messaging mechanism. Messages can communicate with one other using MQTT. Message broker forwards all messages to all clients connected to the topic.

Thing Registry: All connected IOT devices are represented in AWS IOT registry. Each device has unique id and also supports metadata for each device. Devices can be organized into groups and apply permissions to the group.

Authentication: **IOT Devices - X.509 Certificates. Mobile apps – Cognito Identities (Google, Facebook sign in), Web or Desktop / CLI- IAM and federated Identities**.

Authorization: Devices – IOT policies, Users – IAM policies.

Device Shadow: JSON doc representing the state of connected thing (Can be mobile app or Device). It provides persistent representations of your devices.

Rules Engine: Rules are defined on MQTT topics. Rules are used especially when to augment or filter data received from thing and send messages to other AWS services. Rules will trigger rule actions. Rules need IAM policies to perform Actions.

Rule actions can write data to Elastic Cache, Kinesis, Dynamo DB, Machine learning, Cloud watch, S3, SQS, SNS, Lambda. Rules actions cannot write data to Redshift, Aurora.

Athena

Athena is interactive query service that makes it easy to analyse data in S3 using Standard SQL. This enable a capability of hold a data lake of virtually unlimited data and perform queries on them.

Athena uses Hive for DDL and Presto for DML. Athena stores metadata in AWS Glue. Athena supports file formats like CSV, TSV, Json, Text files, ORC, Parquet, AVRO and compressions techniques like snappy, gzip, Lzo. Athena also has JDBC/ODBC interfaces.

Data stored in ORC and parquet format with compression encodings will save cost up to 60%.

Athena integrates with CloudTrial, CloudFront, ELB, VPC for querying logs and Cloud formation for creating named query and work groups, AWS Glue for data catalogue and AWS Quicksight for Visualization.

Data Pipeline

- AWS Data Pipeline that helps to process and move data between AWS compute and Storage services and on-premises data sources.
- Creates a ETL workflow to automate processing and movement of data at scheduled intervals and terminate the resources after its use. Pipe line creates appropriate EC2 instance or EMR cluster in background based on the data load and type of processing. It provisions all resources and terminates automatically.
- Pipeline can also be implemented by lambda.
- Pipeline contains data node, activities, pre conditions and schedules, Destinations can be Dynamo db, Redshift, S3 data node, and SQL data node (RDS or JDBC databases)

AWS Glue

- Fully managed ETL service, it used to move data between various data sources.
- AWS Glue contains Data Catalogue, a metastore. AWS Glue integrates with HIVE for metastore.
- Data crawlers can run through redshift, s2, RDS, databases on EC2 and create schema for underlying data. Crawlers will extract partitions based on how data is organised in S3.
- Glue can be used as event driven ETL pipelines. Glue can create ETL script in python or Scala.
- Job Bookmarks are used to start processing only new data.

AWS ML

- AWS ML only supports Supervised learning, for unsupervised learning we need to use Spark of AWS Sagemaker. ML can be used in fraud detection, customer service, security, predictions.
- AWS ML provides three types of algorithms.

Binary Classification	Multiclass Classification	Linear regression
• Yes/No outcome • **AUC score** is used validate the model accuracy. • Key word examples: Will customer buy a product or not? Will home owner default loan? Is this transaction fraud ?	• Multiple outcomes like book or pen or pencil. • **F1 score** is used to validate the model accuracy • Key word examples: top 50 books every day, recommendations of products to customer based on history.	• Predicting a numeric value • **Root Mean Square Error(RMSE)** is used to validate the model accuracy. • Key words examples: How many products will be sold? What is the worth of house?

- Cross validation is technique for evaluating ML models by training several ML models on subset of available input data to detect overfitting.
- AWS ML can create data sources from RDS, Redshift and S3. AWS ML provides 3 ways to split data- Random split, sequential split, pre-split data.
- AWS ML Limitations: Size of training data – 100 GB, Size of batch prediction input – 1TB, Size of batch prediction input (number of records) - 100 million

AWS Quicksight

- Quicksight is cloud powered visualization tool that can scale to millions of users with impact on performance.

- Data sources: Redshift, Aurora, Athena, RDA (Maria Db, SQL Server, My SQL), databases on EC2 or on-premises, Excel files, log files, extended log files, sale force objects. Quicksight cannot connect to AWS Neptune, Dynamo db, Elastic Search.
- There are two editions: Standard Edition- users invited by any email address or IAM user. Enterprise Edition – It can be integrated with Microsoft AD. It also supports federated logins.
- Quicksight uses SPICE, Super-fast, Parallel, In-memory Calculation Engine. SPICE is engineered to rapidly perform advanced calculations and serve data. Data is imported into SPICE. Each user will get 10 GB initially, can be increased.
- Data preparation is process of transforming raw data for use in an analysis which includes change field names, calculated fields, SQL query, join tables (same data source), change data types, data set checks.
- We can join tables using join interface. Both datasets should be based on same SQL data source. To join tables from different data source, create a join before importing data into Quicksight.
- Stories can be created in Quicksight, it is multiple iteration of analysis and play them sequentially.
- We can create dashboards. Dash boards is read only snapshot of data. It can be shared with other users, they can filter the data in it.
- Quicksight also provide ML insights. Uses Random Cut forest to identify top contributors, significant changes in metrics, to identify outliers and forecasting, to detect seasonality and trends and Anomaly detection.
- Quicksight provides different visual types. It will provide auto graph based on the data. Visual types include Bar Charts (for comparison and distribution – histograms), Line graphs (Changes over time), Scatter Plots (for correlation), Heat maps (for correlations, 2d array with colour based values), pie graphs (for aggregations), Tree maps (Hierarchical aggregations), Pivots (for tabular format of data across multiple dimensions), KPI (key value to target value, forecasted value), Geospatial charts (distribution across maps), Donut charts(like pie charts), Gauge charts (compare value in measure ex: Fuel tank, internet speed), word clouds (repeated words in document), combo charts.
- Quicksight is not used for public deployment. We can use Highcharts and Dj.3 java script libraries for visuals on websites.

Amazon EMR

- EMR contains master nodes, core nodes and task nodes. Master nodes co-ordinates and schedules tasks on task nodes and Core nodes. Code nodes perform actual computing and storage duties. Task node are optional, which adds a computational capacity and stores no data.
- Storage options:
  1)Instance Storage: Directly attached to EC2 instances, they are ephemeral in nature. It is local storage. Used for high IOPS at low cost. Ex: P3 and I3 instance types.
  2)EBS for HDFS: EBS volumes are attached. It is temporary storage.
  3)EMRFS: Uses data directly on S3 without ingesting into HDFS. EMPFS enables to use same data in S3 to multiple Clusters.
  4)EMRFS+HDFS: This storage type is used when same data set is processed multiple times. Data is copied to HDFS from S3 using S3DistCp. It is used high I/O performance.
  5) EMRFS Consistent View: To check for list and read after write consistency for new S3 objects written or synched with EMRFS. Metadata is stored in Dynamo db to keep track of S3 objects. Retry logic if insistency is detected.
- EMR is Single AZ concept because communication between master node, core nodes and task nodes should be reduced.
- Apache Hue (Hadoop User Experience): It is GUI to EMR cluster, easier to manage EMR cluster, view hive metastore, view job status and server logs.
- EMR supports text files, parquet, ORC, AVRO, flat files. S3DistCp is used to combine small files to large files.
- Apache Hive: Data warehouse infrastructure on Hadoop. Uses SQL like language to query called HiveQL. It can be used to process and analyse logs, join very large tables, batch jobs, adhoc interactive queries. We connect to JDBC/ ODBC databases. Partitions are used to increase performance. Hive integrated with Dynamo Db to join hive and dynamo db tables and query them, copy data between dynamo db and hive.
- Apache Hbase: It is No-SQL database like dynamo db, has no limits on item. It is used to handle PB's of data with high throughput and updates. Hbase integrates with S3 for 1) Storage of Hbase store files and metadata on S3. 2). Hbase read replica on S3. 3). Snapshot of Hbase data on S3.
- Apache Presto: It is Open source in-memory fast SQL Querying Engine. Presto can run queries on various data sources. Don't use presto when a queries require lot of memory, and joining large tables, instead use HIVE.
- Apache Pig: Apache Pig is an open-source Apache library that runs on top of Hadoop, providing a scripting language that you can use to transform large data sets without having to write complex code in a lower level computer language like Java. The library takes SQL-like commands written in a language called Pig Latin and converts those commands into Tez

jobs based on directed acyclic graphs (DAGs) or MapReduce programs. Pig works with structured and unstructured data in a variety of formats. Pig integrates with S3 for

a) To submit jobs from AWS console using Pig scripts stored on S3.
b) Directly writing to HCatlog tables in S3.
c) Loading custom jar files from S3 with register Command.

- Apache Spark: Fast Engine for processing large amount of data. It contains four core libraries.

- Spark integrates with Dynamo db, Redshift, RDS, Kafka, Kinesis, Elastic Search.

- HCatlog: It is a tool to access Hive metastore. It has a Rest API and CLI that allow to create tables or other operations.

Security

Kinesis:

Kinesis Data Streams	Kinesis Firehose
• In Transit: SSL Endpoints and HTTPS • At rest: 1) SSE- KMS, 2) CSE – use own libraries • VPC endpoints.	• IAM roles for Redshift, S3, Splunk, Elastic Cache. • At Rest: 1) SSE- KMS • VPC endpoints.

- SSE is enabled on Kinesis Video Streams

SQS:

- In flight: HTTPS.
- At Rest: SSE- KMS, CSE –Own implementation
- VPC endpoints.

IOT:

- Devices – X.509 Certificates, Mobile apps – Aws Cognito identifies, Web and Desktop apps – IAM or Federated Identities, CLI – IAM.
- IOT Policies: can revoke any device at any time. Can be attached to groups instead of individual things.
- IAM: For users and API's. Attach IAM roles to Rules Engine to perform actions.

Dynamo DB:

- In Transit: TLS (HTTPS)
- At Rest: SSE – KMS, S3 –A256
- VPC Gateway.
- Encryption cannot be disabled once enabled and encryption cannot be applied on exiting table, instead create new table with encryption and copy data. Dynamo Db streams doesn't support encryption.

AWS Glue:

- IAM Policies to control actions.
- In Transit: SSL
- Data CatLog: SSE-KMS, resource policies to protect data CatLog resources.
- Connection passwords are encrypted by KMS.
- Data written by Glue: SSE-S3, SSE-KMS, Cloudwatch encryption, job bookmark encryption.

AWS EMR:

- Using EC2 key pair for SSH credentials.
- Attach IAM roles for EC2 to access data in S3/EMRFS, Dynamo Db
- EC2 security groups: one for master node and another one for core and task nodes.
- Supports Kerberos authentication.
- Supports Apache Ranger.
- In Transit: Between nodes – Open source encryption, SSL between cluster and S3, EMRFRS - TLS.
- At rest
  1. EMRFS: SSE – S3, SSE – KMS, CSE – KMS or custom. SSE – C is not supported.
  2. Local disks: Open source HDFS encryption
  3. EC2 Instance Store: Nvme Encryption, LUKS Encryption
  4. EBS Volumes: KMS.
  5. S3: SSE- S3, SSE-KMS, CSE – KMS, CSE – Custom

AWS Elastic Search:

- In Transit: SSL(HTTPS)
- At Rest: KMS
- VPC endpoints.
- IAM or Cognito based for Kibana.
- If encryption is enabled, it encrypts Indices, Automated snapshots, Swap files, all other data in directory, manual snapshots, logs.

AWS Redshift:

- In Transit: SSL
- At Rest: SSE –S3, KMS, HCM (Cloud HSM or on-premises HSM).
- VPC endpoints.
- Cluster Security groups.
- Cluster Encryption.
- Access control Lists
- Use Views to restrict data.
- Enable VPC routing forces redshift to use VPC for COPY and UNLOAD commands which in turn make sure that all traffic appears on VPC flow Logs.

AWS Quicksight:

- Standard Edition: IAM and email based accounts.
- Enterprise Edition: Microsoft AD and Federated logins. Encryption at rest.
- Row Level Security.

In transit encryption for all services.	
SSL(HTTPS)	TLS
Redshift, Elastic Search, Glue, Kinesis, SQS	Athena, EMR, Dynamo DB, Quicksight.
VPC Support	
Not supported	Supported
IOT, Glue, EMR, Athena , Quicksight	Kinesis, SQS, S3, Dynamo Db, Lambda, Elastic Search, Redshift.

General Information which helps in eliminating the options.

- Apache Flink is open source streaming data flow engine that can be configured to un real time stream processing with high throughput. Flink supports event time semantics for out of order events, exactly once semantics, back pressure control and API optimized for writing both batch and streaming data.
- Apache Phoenix is a used for OLTP and operational analytics which allows to use SQL and JDBC AOI to work with an Apache Hbase backing store.

- Apache Tez is framework for creating complex directed acyclic graph of tasks for processing.
- Apache Zookeeper is centralized service distributed configuration service, synchronization service, and naming registry for large distributed systems.
- Apache Sqoop is a tool for transferring large amounts of data between S3, Hadoop, HDFS and RDBMS databases.
- Apache Zeppelin is web based notebook that enables data driven interactive data analytics and collaborative documents with tools such as SQL and Scala.
- Jupyter Notebook is open source web application to create and share live code, visuals.
- AWS Comprehend uses Natural Language Processing to extract insights from contents of document.
- AWS Rekognition analysis images and video to identify any object, people, text or scenes. It is highly accurate with face recognition.
- AWS Polly is used to convert Text to speech.
- AWS Elastic Transcoder is used to convert media files in S3 into multiple format supported by playback devices.
- AWS Transcribe is uses Advanced ML to recognize speech in audio files to transcribe them into text.
- AWS Neptune is fast reliable full managed graph database service to store billions of relationships.
- AWS Lex is used for building conversational interface for applications using voice and text like Chabot's.
- S3 storage gateway connects on premises infrastructure with cloud. It has two types
    1. File Gateway: It supports a file interface into S3, using this we can store and retrieve objects in S3 using NFS and SMB.
    2. Tape Gateway: It provides a cloud backed virtual tape storage, it is deployed to on premises. This provides virtual infrastructure that scales seamlessly.
    3. Volume Gateway: It provides cloud backed storage volumes that can mount as device on-premises application server. It can be
        a) Cached Volume: we can store a copy in s3 and retain a copy of frequently accessed data subsets locally.
        b) Stored Volumes: If we need low latency to entire dataset, we can configure on-premises gateway to store all the data locally and have point in time backups to S3.
- S3 cross region replication: To enable bucket replication in same or cross region in same account or different account.
- To compare objects is S3, we use Etags, to ensure files doesn't have duplicated or have same content.
- RDS doesn't allow authorization using IAM policies.
- Direct connect only for connecting on-premises data centre to AWS VPC.
- When Elastic Cache is in VPC, we allow Kibana from outside by SSH tunnel with access port 5601, reverse proxy between Brower and Elastic Cache.

Please mention if anything I've missed. This notes would work for Aws Data Analytics – Speciality certification with few new changed.

Thank you.