AWS Big Data / Data Analytics Specialty Exam Study Materials

By Yitaek Hwang

AWS Kinesis **2**
- Data Streams 2
- Firehose 4
- Kinesis Data Analytics 4
- Kinesis Video Streams 5

EMR **6**
- Hadoop Ecosystem 6

Data Pipeline **8**

DynamoDB **8**

QuickSight **9**

AWS Glue **9**

Migration Services **10**

Storage **10**

Redshift **10**

Athena **12**

Machine Learning **13**

AWS Kinesis

Comparison of Kinesis Data Streams:

- Kinesis Agent: logs collection, **watches file changes**
- KPL: batch, buffer, auto-scale
- Kinesis Data Streams API: real time ingestion
- Kinesis Connector Library: connect Data Streams with other AWS services
- KCL: read and process data

Data Streams

- Synchronously replicates data across 3 AZs
- Key Concepts:
  - Stream: set of shards
  - Shard: sequence of data records
  - Record: unit of data composed of a sequence number, partition key, and data blob
    - Partition key: used to segregate and route records to different shards
    - Sequence number: unique ID
- Store stream's records for 24 hours (up to 7 days). **Max size of 1 MB and up to 1000 PUT/s**. A shard provides 1MB/sec (writes) input and 2 MB/sec (reads) output
- KPL cannot read log file (use Kinesis agent instead). **KPL maxed out at 1000 records/s and 1MB throughput per shard. If the record is bigger than 1MB, us Streams API instead up to 5MB.**
- KPL user record vs. Data Streams record:
  - User record: blob of data that has particular meaning to the user
  - Data streams record: instance of the record that contains partition key, sequence number and blob of data
- Number of shards = max (writes / 1000 , reads / ~~2000~~) in KB
- **Kinesis Agent:** mostly used for log collection. It monitors a set of files and sends new data to the stream. The agent can handle file rotation, checkpointing, and retries.
- **KCL**: pre-built library to read and process data (load balancing, coordination of distributed services, and fault-tolerance). It uses a PUSH (vs. PULL for Data Streams API)
  - Generates a worker and a record processor. One worker maps to 1+ record processors. One record processor maps to one shard and processes records from that shard.
  - **Creates an DynamoDB table to track and maintain state information (resharding, sequence number checkpoints)**
  - Checkpoints processed records

    - Enhanced fan-out provides up to 2 MiB/sec of data per shard
- **Kinesis Data Streams API** is used for real time ingestion (vs. KPL, which can incur additional processing delay up to RecordMaxBufferedTime). Also useful if the data size is too big for KPL (5MB)
- **Kinesis Connector Library**: pre-built library to connect Data Streams with other AWS services and 3rd party tools (DynamoDB, Redshift, S3, an ES)
- Resharding: process used to scale data streams using splits or merges (**can only do one resharding at a time**). Parent shards don't disappear when the reshard occurs (data records that were in the parent shards remain in those shards)
- **KPL concepts (Batching)**: both are turned on by default
    - **Aggregation**: use one Kinesis Data Streams record to include multiple records that will share the **same partition key** to maximize 1MB writes (increases the number of records sent per API call)
    - **Collection**: Batch multiple streams records in a **single HTTP request** via PutRecords API operation
- Use Future objects to validate UserRecords
- Encryption:
    - Server-side encryption is always enabled on Kinesis Video Streams
    - SSE-KMS can be enabled using CMK managed by AWS or CMK managed by customer
- Monitoring:
    - Shard-level metrics are sent to CloudWatch every minute (not enabled by default)
    - **KPL metrics:**
        - Levels: NONE, SUMMARY, DETAILED
        - Granularity: GLOBAL, STREAM, SHARD
    - **KCL metrics:**
        - KCL-application metrics: aggregated across all KCL workers as defined by CloudWatch namespace
        - Per-worker metrics: aggregated across all record processors consuming data from a Kinesis data stream
        - Per-shard metrics: aggregated across a single record processor
    - **Kinesis agent:**
        - Bytes sent, number of records attempted, number of returned failures / service error
- Retry:
    - When KPL records are added to a stream, a record is given a timestamp and added to a buffer with a deadline set by the RecordMaxBufferedTime, which sets the buffer priority.
    - Records that fail are automatically added back to the buffer with new deadline set based on min(½ RecordMaxBufferedTime, TTL)

    - **TTL: addresses durability of the stream. Records that do not get put within this limit are failed. Useful if application cannot or does not wish to tolerate late records**

Firehose

- Fully managed, auto-scaling data delivery service to Kinesis Analytics, S3, Redshift, ES, or Splunk (**NOTE: DynamoDB is not supported**)
- Can use CreateDeliveryStream, Firehose API, or Kinesis Agent
- Can compress via GZIP, ZIP, and SNAPPY (only GZIP for Redshift)
- For ETL, use Lambda to transform prior to delivery
    - General data transformation
    - Apache Log → JSON/CSV
    - Syslog → JSON/CSV
    - CloudWatch
- Buffers data between 60-900 seconds in 1-128MB batches (for S3 and 100MB for ES). Buffer size is applied prior to compression
- Insert to Redshift is actually a COPY from S3. S3 is also used as backup while delivering to ES
- Limits: 2000 transactions/s, 5000 records/s, 5MB/s
- Can convert JSON to Parquet and Apache ORC prior to storing into S3
- Encryption: encryption at rest only works with Kinesis data streams

Kinesis Data Analytics

- Process and analyze streaming data using SQL for time series analysis and real time dashboards
- ROW TIME: timestamp when data was inserted into the row of the stream
- INGEST TIME: timestamp when record was added to the streaming source
- **Windows queries**
    - **Stagger**: It's best suited for groups of data that *arrive at inconsistent times*. Windows open when the first event matching the partition key arrives (not on a fixed time interval). The windows close based on the age specified from the time the window opens. Stagger is better than tumbling windows to address partial results.
    - **Tumbling:** using *distinct time-based windows* that open and close at regular intervals (non-overlapping, aggregations using GROUP BY).
    - **Sliding:** aggregates data continuously using a fixed time or rowcount interval (define a time-based or row-based window using WINDOW clause)
    - Continuous

Kinesis Video Streams

- Provides APIs for creating and managing Kinesis video streams as well as reading and media data to a stream. KPL is used to extract data from media sources and upload to Kinesis video stream.
- Each stream stores: (1) copy of media metadata, (2) fragment, (3) Kinesis Video Streams metadata (server and producer side timestamps)
- Support HLS for live playback or viewing archived video.
- GetMedia API is for real-time
- Stream live video, but is **NOT used for conversion** (use Amazon Elastic Transcoder instead)
- Metadata:
    - Nonpersistent: adhoc labeling (e.g. motion = true)
    - Persistent: continuing need (e.g. adding location)

EMR

- Core nodes stores data in HDFS vs. task nodes (*only runs tasks and no storage*)
- Metadata on EMR is stored in Glue Data Catalog (e.g. Presto). It can also use RDS or Aurora to store.
- SSE-KMS and LUKS encryption are supported
- Use S3DistCp to copy data from S3 to HDFS to improve performance and optimize network costs
- Use Bootstrap Actions to install 3rd party software
- Consistent view allows EMR clusters to check for a list and read-after-write consistency
- Security:
  - At rest: open-source HDFS or LUKS encryption
  - In transit: EMRFS <-> S3 TLS is enabled automatically
  - At-rest (EMRFS): SSE-S3, SSE-KMS, CSE-KMS, or CSE-Custom
- Data file formats:
  - Parquet (column-based) - optimized for read-heavy
  - ORC (column-based) - optimized for read-heavy
  - Avro (row-based format) - optimized for write-heavy
- Compression
  - Gzip: high compression
  - Snappy: good for frequently accessed data
  - Bzip2: splittable

Hadoop Ecosystem

- Resource Management:
  - **YARN:** Yet Another Resource Negotiator
- MapReduce:
  - **Tez**: framework for creating complex DAGs (alternative to MapReduce)
- Storage:
  - **HDFS:** Hadoop Distributed File System
  - **HBase**: NoSQL, distributed database modeled after BigTable. Integrates with Hive.
  - **Phoenix**: OLTP and operational analytics that works with HBase
- Data Transfer:
  - **Flume:** unstructured and semi-structured data (log)
  - **Sqoop**: structured data
- Scheduling:
  - **Oozie:** job scheduling

- Management:
    - **Zookeeper:** managing clusters
    - **Ambari**: provision, monitor, and maintain cluster
    - **Kerberos:** authentication
- Scripting:
    - **Pig**: similar to Hive, but Hive supports **SQL-like queries** whereas Pig uses Pig Latin
- Machine Learning:
    - **Spark**: utilizes **in-memory** caching, and optimized query execution
    - **Mahout, Spark MLlib**: ML libraries
- **Presto**: distributed SQL query engine - used by Athena to execute DML (data manipulation). **Good for interactive queries**.
- **Hive**: data warehouse and analytics package running on top of Hadoop. Good for offline reporting, transformation, and analysis of large data sets - used by Athena to execute DDL (data definition) that create and modify schema. **Good for batch operations**.
    - Interactive: run Hive scripts on master node ad hoc
    - Batch: Hive scripts stored on S3 and references at the start of the cluster
- **Apache Drill**: SQL on Hadoop (similar to Hive)
- **Impala**: similar to Hive, but uses massively parallel processing (MPP) engine found in RDBMS instead of MapReduce to query data in HDFS or HBase quickly. Use for fast, interactive queries vs. Hive is good for ETL workloads on large datasets (not constrained by memory resources)
- **Flink**: streaming dataflow engine for realtime stream processing

Data Pipeline

- Service used to automate the movement and transformation of data.
- Pipeline components: business logic
- Pipeline components creates a set of actionable instances
- Pipeline hands the instances out to task runners to process

DynamoDB

- (Desired RCU/3000 RCU) + (Desired WCU/1000 WCU) = # of partitions needed
- Capacity Units:
    - 1 RCU = 1 read/s (consistent) or 2 reads/s (eventual) for items up to 4KB
    - 1 WCU = 1 write/s for items up to 1KB
- Turning on encryption is only possible upon creation (i.e. can't encrypt existing table)
- Global tables are supported for multi-region, multi-master databases
    - Must have same partition keys
    - Global secondary indexes must have the same partition and sort key
    - DynamoDB streams must be enabled
    - Must have same write capacity (read capacity can differ)
- Index: partition key and sort key
- Scan operation reads an entire page (by default 1MB), so set a smaller page size to reduce read costs
- Security
    - Encrypts data at rest by default (stores keys in KMS)
    - When creating a new table, customer master key options (can switch):
        - AWS owned CMK - default
        - AWS managed CMK - key is stored in your account and managed by AWS KMS
        - Customer managed CMK - fully managed by customer
- Random suffixes improves writes and calculate suffixes improves reads
- Best Practices
    - **Global secondary index**: index with a partition key and a sort key that can be different from those on the base table. It is global in that index can span all the data in the base table across all partitions.
    - **Local secondary index: index with the same partition key as the base but a different sort key. (Limit of 10GB)**
    - Sparse Indexes: DynamoDB writes a corresponding index entry only if the index sort key value is present in the item. If the sort key doesn't appear in every table item, the index is said to be sparse.

    - **GSI Overloading**: same attribute in different items can contain different types of information → allowing you to use that attribute as sort key and make different queries using that single global secondary index
    - **GSI Sharding**: enable selective queries across the entire key space, write shard by adding an attribute containing random 0-N value to every item on the global secondary index partition key → allows a scatter read using a sort condition on the composite key
- Capacity:
    - Burst capacity: DynamoDB reserves a portion of unused capacity for usage spikes.
    - **Adaptive capacity**: enables continued read/write to hot partitions without being throttled (provided that traffic does not exceed table's total provisioned capacity or partition maximum capacity)

QuickSight

- To join tables from different data sources, create the join before importing.
- Allowed configuration join type can be inner, outer, left, or right
- Cannot pull directly from Neptune, DynamoDB, and ES (but can read XLSX/ELF/CLF/CSV/TSV from S3)
- Login:
    - Active Directory
    - Federated login
    - Email
    - (No IAM)
- Chart Types:
    - Combo chart: clustered bar / stacked bar chart (line and column)
    - Tree maps: rectangular box chart

AWS Glue

- Glue Data Catalog: crawlers scan data, classify, and extract schema information and store metadata automatically
- Glue auto generates Scala or PySpark scripts with Glue extensions that use and modify ETL operations via Glue ETL operations
- **When crawler detects multiple directories and finds a schema that's similar, crawler may treat them as partitions instead of separate tables. Add each table's root directory as a data store for the crawler to help it discover individual tables.**

Migration Services

- Database Migration Service: one-time migrations to migrate relational, NoSQL, data warehouses, and other types of data from on-prem to cloud.
- DataSync: data transfer that automates moving and replicating data between on-prem storage and AWS storage over Direct Connect or internet.
- Migration Hub: place to discover existing servers, plan migrations, and track the status of each application migration
- Schema Conversion Tool: convert existing database schema from one database engine to another

Storage

- Volume Gateway: mount iSCSI devices from on-prem to cloud with cached volumes (store data in S3 and retin copy locally) or stored volumes (store data locally and do async back ups of snapshots to S3)
- File Gateway: file interface to S3 (NFS or SMB)
- Tape Gateway: durable archive backup data with virtual tape infrastructure

Redshift

- AQUA - DAX equivalent cache for Redshift
- Loading data via S3 and DynamoDB is fast as it uses parallel loading vs. using ODBC/JDBC SQL insert
- COPY is available from S3, EMR, DynamoDB, or SSH-enabled host. S3 also supports INSERT INTO via Redshift Spectrum
- Data Pipeline and Glue also provide ways to easily load data
- Upsert: Redshift doesn't support a single merge statement (update or insert / upsert) from a single data source. Load the data into a staging table and then join the table with UPDATE then INSERT
- STL system tables keep logs and provide a history of the system
- STV system tables contain snapshots
- Distribution
    - Key: distributes the rows according to the values in one column
    - All: makes a copy of the entire table in every compute node (small or rarely-updated data sets)
    - Even: round-robin distribution (good for no joins)
    - Default: ALL → EVEN as tables get large

- Sort key
    - **Compound:** specifies precedence among the sort key columns (checks first -> tie break using second key). Good performance with 6 or fewer keys
    - **Interleaved:** treat each column with equal importance
    - If you frequently join a table, specify the join column as both sort and distribution key to do a sort merge join instead of a slower hash join
- Security:
    - Set data set rules for row-level security
    - Apply row-level permissions by using a file or query that contains data set rules
    - Encryption: not enabled by default
    - List of logs:
        - Connection: auth attempts, connection attempts
        - User: changes to database user definitions
        - Activity: log of queries
- Streaming data is delivered to S3 bucket first and Firehose issues Redshift COPY command. When backup is enabled, untransformed incoming data can be delivered to a separate S3 bucket and errors to another bucket
- Maintenance
    - Svv_table_info: gives a high level system view
    - **ANALYZE: update the statistical metadata for optimal queries (or use COPY with STATUPDATE on)**
    - VACUUM: re-sort data added to non-empty tables and recover space from delete operations
    - Stl_alert_event_log: flags potential performance concerns
    - Stl_wlm_query: shows queing times for the cluster
    - **STL_LOAD_COMMITs**: verify expected files were loaded
- Node types:
    - RA3: flexible scale of compute and storage
    - DS3: for large storage (HDD)
    - DC2: for compute (SSD)
- Redshift does not enforce unique, primary-key, and foreign-key constraints
- Workload Management (WLM):
    - can configure up to eight query queues and set number of concurrent query limit in each
    - Short query acceleration (SQA) prioritized selected short-running queries in a dedicated space
- Updating:
    - Make use of staging table to most efficiently add new data
- Evaluating Query Plan:
    - **DS_DIST_NONE and DS_DIST_ALL_NONE = no distribution was required for that step (all join are collocated)**
    - DS_DIST_INNER = inner table is being redistributed to the nodes

    - DS_DIST_ALL_INNER = entire inner table is redistributed to a single slice (because the outer table uses DISTSTYLE ALL)
    - DS_BCAST_INNER / DS_DIST_BOTH are also not good. This usually means the tables are not joined on their distribution keys.
- Encryption:
    - Redshift can switch to encrypted mode. During the migration process, the cluster is read only mode and status is set at resizing. Cross-region snapshot copy must be disabeld before changing encryption. For HSM, you need to create a new and migrate.
    - To enable cross-region snapshot, need to configure snapshot copy grant for master in destination and create a new KMS key in the new region
- Snapshots: automatic snapshots saved to S3 and restore creates a new cluster
- Extract data via UNLOAD
- For cross-table data copying, use insert or create table
- Best Practices:
    - Number of files should be a multiple of the number of slices in your cluster
    - Optimal size of files should be between 1-125MB
    - For time series data, use UNION ALL to hide the fact that the data is stored in different tables
    - Use DROP TABLE instead of DELETE and VACUUM to reclaim space
    - To minimize the number of commits in an ETL process, surround with transaction handling (begin/end statement)

Redshift Spectrum

- Querying from S3 in the original format is possible if the data is in the same region
- Compression types supported: gzip, snappy, bzip2

Athena

- SerDe (Serializer/Deserializer): way in which Athena interacts with data in various formats
- Only pay for the queries (create table as select doesn't count for billing)
- Permissioning:
    - Athena actions
    - S3 locations where underlying data is stored
    - Resources stored in AWS Glue Data Catalog (databases, tables)
    - Encrypted metadata

Machine Learning

- **Comprehend**: NLP to extract insights about the content of documents (**entity recognition**)
- **Rekognition**: image analysis including **facial recognition**
- **Polly**: speech to text
- **Lex:** chatbot capability (think aLEXa)
- Amazon ML supports both batch and realtime
- No Unsupervised learning → use Spark