AWS Data Analytics Specialty

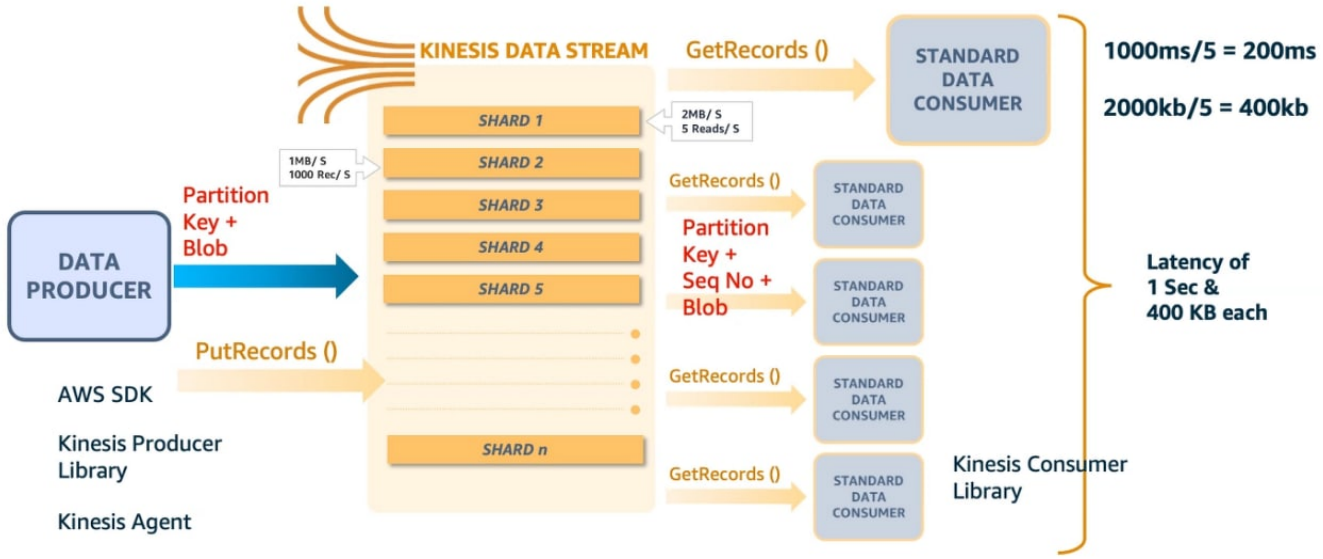
- AWS practical workshops
- AWS Partner practice exams
- Official AWS exam guide
- Official AWS exam sample questions
- Tutorialsdojo AWS Analytics cheatsheets
- Tutorialsdojo exam study path

1. Data collection

Stream	**Firehose**
Managed service but shards need configuration	Fully managed
Real-time	Near real-time

Stream	Firehose
Provides storage	Does not provide storage
200ms latency for normal, 70ms for enhanced	min buffer time 60s
Max data blob: 1MB	Max data blob: 1MB
Data retention from 1 to 365 days	No data retention
Manual scaling	Automatic scaling
Prod: Agent, SDK, KPL, Lambda, Spark Streaming	Prod: Agent, KDS, DirectPUT (not CloudWatch Agent)
Cons: KDF, KDA, KCL, Lambda, Spark Streaming	Cons: KDA, S3, Redshift, Opensearch, HTTP endpoint, Splunk, MongoDB

1.1 Kinesis Data Stream



A stream is an ordered (fifo) stream of data, composed of shards, in which each record is uniquely identifiable. A record is made of a partition key and the data blob (up to 1mb). the partition key determines which shard the value goes to. when in the shard, a sequence number is added (where it was in the shard).

The user has to design a partition key and throughput, according to that shards get allocated. A bad partition key -> hot shards.

Types of setup:

- Standard consumer
  - Avg latency 200ms
  - Throughput 2MB/s/shard shared for all consumers
- Enhanced fan-out
  - Avg latency 70ms
  - Dedicated throughput of up to 2MB of data/s/shard/consumer

Once data is inserted into kinesis, can't be deleted (immutable). Data that shares the same partition goes to the same shard (ordered).

Capacity mode:

- Privisoned mode
  - Choose number of shard provisioned, scale manually or with API
  - Each shard gets 1MB/s in, or 1k records/s
  - Each shard gets 2MB/s out (classic or enhanced fan-out consumer)
  - You pay per shard provisioned per hour
- On-demand mode
  - No need to provision or manage capacity
  - Default capacity provisioned: 4MB/s or 4k records/s
  - Scales automatically based on observed throughput peak during the last 30 days
  - Pay per stream per hour and data in/out per gb

Partition by session ID, not timestamp, since date is associated by time and not to a specific user.

Producers

- SDK
  - Use case: low throughput, hig latency, simple API, Lambda
  - PutRecord: for one, PutRecords: for many, uses batching, increases throughput, fewer HTTP requests
- KPL (kinesis producer library)
  - C++/Java library
    - If errors in the Java application result in data loss, modify the Java application code to use idempotent processing by pointing the shard iterator to the shard position before the application error occurred.
  - Use case: high performance, long-running projects. Automated and configurable retry mechanism
  - Adds a delay of up to RecordMaxBufferedTime. Larger values of this variable result in higher packing efficiencies and better performance. Apps that cannot tolerate this additional delay need to use the SDK directly
  - Sync and async API (better performance with async)
  - Compression can be implemented, but by the user
  - Batching enabled by default, it increases throughput and decreases cost
    - Aggregation puts several records together (but <1MB), increases latency, but increases efficiency
- Kinesis Agent
  - Java based agent that can be installed in the app, built on top of KPL, monitors log files and sends them to data streams
  - But for EC2, no need for agent you can use Cloudwatch logs
    - You can use subscriptions to get access to a real-time feed of log events from CloudWatch Logs
    - Installing Kinesis agent to the EC2 instance is unwarranted since there is already a CloudWatch Logs integration that can deliver the logs.
  - Writes from multiple dirs and writes to multipls streams

Consumers

- SDK:
  - Records up to 10MB data or up to 10k records, then throttle for 5s, then again
  - Max 5 GetRecords API calls/s/shard because of the 200ms latency
  - The stream has limits about how many MB/s, and how many records/s per shard. Example: the stream can generate 2MB/s, or 5reads/s per shard. If we have 5 consumers, then each consumers has 400kb/s read capacity
- KCL (Kinesis client library): not AWS managed
  - Java-first, but exists for other languages
  - read records from Kinesis produced by KPL
  - Uses DynamoDB for coordination and checkpointing, one table per KCL app. If throughput is low, it means that DynamoDB is underprovisioned
  - Already has the logic for parent-first-reading after resharding
  - Not for SQL
- Lambda
  - it can de-aggregate from KPL
  - can be used to store data in s3, dynamodb or to run lightweight ETL to Redshift, Opensearch
- **KDS cannot deliver data directly to S3 or Opensearch**

Double data issues:

- Reading same data twice can happen if consumers retry. To fix it, make your consumer app idempotent (that there's no side effects for reading the same data twice), and that try to handle duplicates in the final destination
- Writing same data twice issue: network timeouts might create duplicate records, and they will get unique sequence id even though it's the same data. To fix it, embed unique record ID in the data to de-duplicate on the consumer side

If a record arrives late to the app during stream processing, it's written into the error stream.

Resharding

After reshard, configure your consumers to read entirely from the parent until there are no new records to avoid reading data for a particular PK out-of-order

- Adding shards
  - more stream capacity, more costs
  - can be used to divide a hot shard (if we divide, then there's 1MB/s per shard)
- Merging shards
  - less stream capacity, less costs
  - can be used to group two shards with low traffic

To increase capacity of stream, first use KDS in on-demand mode, and then start splitting shards.

Autoscaling

Not native to Kinesis, you can use it with *UpdateShardCount*.

Resharding can't be done in parallel, you have to plan capacity in advance. You can only perform one resharding operation at a time and it takes a few seconds, redoubling shard would take a lot of time. There's also a limit on how fast you can scale up and down.

> Errors and their meanings

- ProvisionedThroughputExceededException errors: increase shards, retried with backoff, make a better PK, move to enhanced fan-out
    - Same error name for DynamoDB, in this case increase WCU
- RecordMaxBufferTime error, increase batch efficiency by delay
- ExpiredIteratorException KCL error, increase WCU of Dynamodb

Performance issues:

- If bad performance on write, use random partition keys, and increase shards. retry and exponential backoff won't help
- If GetRecords.Latency is increasing, increasing shards is the cheapest solution. You can also implement enhanced fan-out but that's more costly

> Security

- encryption in flight using https endpoints
- encryption at rest using kms
- encryption on client side can be implemented
- vpc endpoints are allowed, so that an ec2 instance in a private subnet can access the kinesis stream

1.2 Kinesis Data Firehose

Near real-time streaming service with no storage, min 60s latency and can use Lambda for transformation.

S3 as destination:

- By default, firehose-s3 default prefix is already based on year,months,day,hour. if we have many devices, we want to use custom prefix based on device and date (in our example, once a day)
- If the queries are limited to one day's logs, rotate directories daily

Firehose accumulates records in a buffer, which is flushed based on time and size rules. If size, if you make the rule of 32mb, then if that buffer size is reached, it's flushed. for time, if you set 2mins, it's flushed every 2mins. Firehose can automatically increase the buffer size to increase throughput. For high throughput scenarios, use buffer size limit. For low throughput, use buffer time limit.

- If the Firehose delivery stream has scaled, there might be more files in the destination
- When data delivery to destination is falling behind data writing to a delivery stream, Firehose raises buffer size dynamically to catch up and make sure that all data is delivered to the destination. The size of delivered S3 objects might be larger than the specified buffer size
- Firehose delivers smaller records than specified if compression is enabled on the Firehose delivery stream

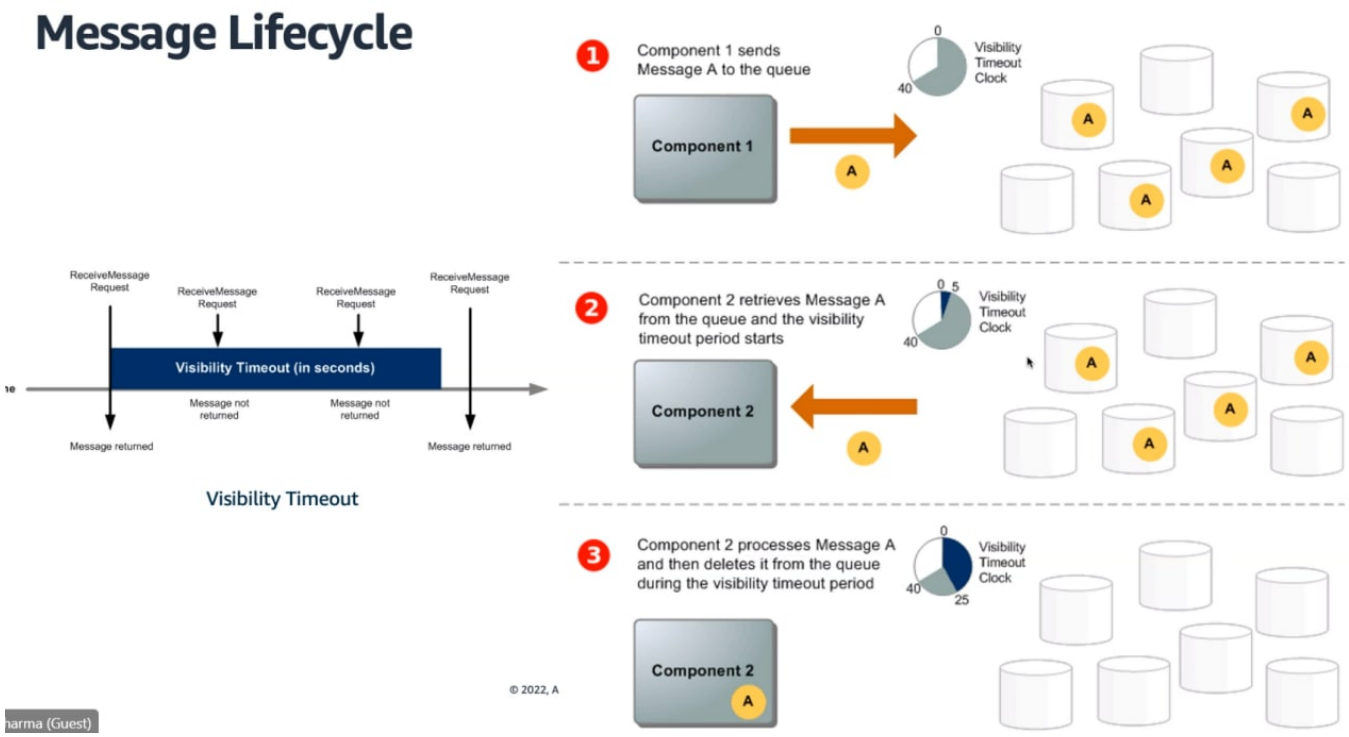
Firehose can be used to aggregate logs from different AWS accounts and receive their loggins in a centralized logging AWS account. Set up Kinesis Data Firehose in the logging account and then subscribe the delivery

stream to CloudWatch Logs streams in each application AWS account via *subscription filters*. Persist the log data in an Amazon S3 bucket inside the logging AWS account

Security:

- Server-side encryption supported with KMS
- VPC endpoints supported

1.3 SQS



Secure, durable and available hosted queue to integrate and decouple distributed components. Can be another ingestion service

- Standard:
  - unlimited throughput
  - at-least once delivery
  - best effort ordering
- FIFO:
  - high throughput upto 3k messages/s, per API method (with batching) or up to 3k API calls/s per API method (without batching)
  - Exactly once processing, no duplicates
  - maximum of 10 transactions per operation. With e.g. 8 transactions per operation, you can support up to 2400 transactions/s

Data retention is 3 days default, max 14 days. Low latency, <10ms on publich and receive. message size is 256kb max. SQS only allows one consumer.

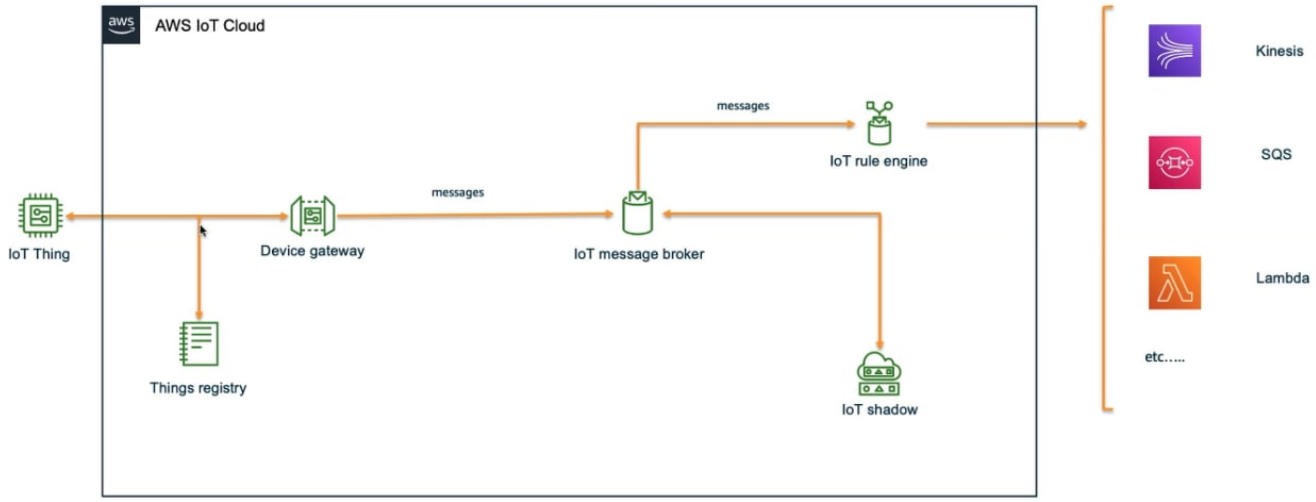
Security:

- Encryption in-flight using HTTPS
- Server-side encryption with KMS

- SQS queue access policy
- Client-side encryption is allowed, but must be implemented manually
- VPC endpoint provided through an interface

1.4 IoT

- Device gateway: entry point for IoT devices to AWS
- Message broker: pub/sub, messages published to topics, forwards messages to all the clients connected to the topic
- Thing registry: represents all connected devices represented by their ID: supports metada for each device, can group devices together
- Device shadow: JSON document to represent state of a connected thing
- Rules engine: when a rule is triggered, an action is taken e.g. write data to Kinesis, SQS or so. IAM role needed
- Greengrass: bring the compute layer to the device directly
- IoT core: to process MQTT messages from the devices



Security:

- IoT policies
- IAM policies, to control access to IoT API
- Roles can be attached to rules engine to perform their actions

1.5 Data migration

- Data migration service (DMS): to migrate on-prem or EC2 databases into aws
    - With Schema conversion tool, you can convert the db schema from one engine to another
    - To check that the data was migrated accurately, use DMS data validation on the migration task so it can compare the source and target data for the DMS task and report any mismatches
    - Premigration assessment helps identify any problems that might prevent a migration task from running as expected
    - Useful when you have an on-prem db and you want a read replica, which can't be instantiated from an on-prem db. Use DMS to migrate db to RDS or similar
    - Can be used to migrate RDS into Redshift. KDS works too but requires custom development

    - When a migration task that is replicating data to Amazon Redshift has an issue applying a batch, AWS DMS doesn't fail the whole batch. AWS DMS breaks the batch down and switches to a one-by-one mode to apply transactions. When AWS DMS encounters the transaction that caused the batch to fail, AWS DMS logs the transaction to the awsdms_apply_exceptions table on the Amazon Redshift target. Then, AWS DMS applies the other transactions in the batch one by one until all transactions from that batch are applied onto the target. Finally, AWS DMS switches back to Batch Apply mode for a new batch and continues to use Batch Apply unless another batch fails
- Storage gateway: hybrid storage that connects on-prem to AWS. ideal for backup, bursting, tiering, migration
- DataSync: 5x faster file transfers than open source tools, good for migration data into AWS storage services or moving between cloud file systems
    - Automatically encrypts at-rest and in-transit, performs data integrity checks
    - To move data from on-prem to AWS, deploy a DataSync agent on-prem and replicate the data to S3 or whatever
- Direct connect (DX): private connection between on-prem and aws with dedicated fiber optic, no ISP provider required. increased bandwidth
    - dedicated connection: 1Gbps, until 100Gbps
    - hosted connection: 50Mbps, until 10GBps if you order from approved aws direct connect partners
- Snow family: physical pre-packaged hardware box that gets delivered to your on-prem system
    - Snowcone: 2GB box, lightweight device used for edge computing, storage and data transfer. use it offline or connect via internet with aws datasync to send data (agent is preinstalled in snowcone)
        - Storage capacity: 80TB usable, migration size up to petabytes
    - Snowball edge: storage optimized with ec2 compute functionality. also available: compute optimized and compute optimized with GPU
        - Good option when data is distributed across many locations
        - Can transfer up to 9PB
        - Can only migrate to S3 standard, then you can create lifecycle policy to move to Glacier
    - Snowmobile: for huge data transfers, they send a truck instead of a box
        - Can transfer up to 100PB, recommended for more than 10PB. for less, Snowball edge
        - Can migrate data directly to Glacier

1.6 MSK

- MSK documentation
- Kinesis & Kafka for fraud detection workshop

Managed Streaming for Apache Kafka is a managed Apache Kafka to ingest and process streaming data in real time. It can only span 1 region, in several AZs.

MFK	**KDS**
Max message size: 100MB	Max message size: 1MB
Topics and partitions	Streams and shards
Only add partition to topic	Can split and merge shards

MFK	KDS
Plaintext or TLS in-flight encryption	TLS in-flight encryption

Producers write to the cluster (you have to write code for this), then the data gets written to the topic and replicated into the other topics, and then consumers poll from topic (write code). Consumers can be KDA, Lambda.

The MSK cluster is composed of keeper and broker, broker is where you keep partitions.

Security:

- Optional in flight encryption using TLS between brokers, and between clients and brokers
- KMS for EBS for at encryption rest
- Security groups for clients to ensure network security

You are charged for the following:

- Every Apache Kafka broker instance
- The amount of storage you provide in your cluster

MSK serverless: for intermittent non-constant workloads, no need to manage or scale cluster capacity. Pricing depending on cluster, partition, and storage

MSK connect (like consumer) works with other 3rd party services, we can use plugin. you can deploy any kafka connect connectors to MSK connect as plugin like S3, redshift, openserach. SK connect workers can poll topic data from SMK cluster, and write to S3. No need to manage infrastructure.

Source connectors can be used to import data from external systems into your topics. with sink connectors, you can export data from your topics to external systems.

2. Storage

2.1 File formats

- Parquet
    - Columnar format for Hadoop, stores nested data structures
    - compressed and splittable
    - max recommended file size: 250MB
    - Good option to compress JSON
    - Good for improving queries for tables
    - Use PySpark to concatenate Parquet files, S3DistCP doesn't support it
- ORC
    - columnar, good for improving queries for tables
    - compressed and splittable
    - Good option to compress JSON
    - Only supports Hive and Pig
- Avro
    - row-based format for Hadoop
    - Stores in JSON format

    - To optmize files in Hadoop, use AVRO to compress and then uncompress into 64MB chunks, which is the default HDFS chunk size
- Csv
    - Row-based
- Gzip
    - compressed row-based

2.2 S3

- Latency is between 100-200ms
- Min. 3.5k put/copy/post/delete and 5.5k get/head requests per second per prefix per bucket
- Strong consistency for all operations: GET, PUT, LIST, all operations with metadata
- S3 Select
    - To get just a subset of data, cheaper than Athena
    - Can do select from compressed data
    - Byte Range Fetch: fetch a byte-range from an object, like 250 bytes
    - ScanRange: scan a subset of an object, specify a range of bytes
- Access point: you can create unique access control policies for each access point to easily control access to shared datasets or different prefixes in the bucket
- S3 object Lambda: you can change the objects before it's retrieved by the caller application, without creating another object. for that, we need 1 bucket, an access point and an object lambda access point
- Cross region replication: to reduce latency with other regions and to keep the data as up to date as possible
- S3 and S3 one-zone IA have the same read performance
- To improve read performance, add random string prefixes to objects

By default, an S3 object is owned by the AWS account that uploaded it. So if you upload a file to another account, this account owner will not implicitly have access to it.

Glacier

- Glacier Select: to query cold data stored in Glacier as fast as possible. Cannot be used for compressed files. If no urgency, it's cheaper to restore data to S3 and then S3 select
- Glacier vault lock policy prevents anyone from deleting data in Glacier
- Glacier vault access policy prevents access to data in Glacier

Encryption

3 methods for encrypting objects in S3:

- SSE-S3: objects are encrypted using keys handled and managed by S3
    - AES-256 encryption type, must set header "x-amz-server-side-encryption": "AES256"
- SSE-KMS: keys managed by KMS
    - Advantages: user control and audit trail. Header: "x-amz-server-side-encryption": "aws:kms"
    - You can use different KMS keys to encrypt the objects, and then you can grant users access to specific sets of KMS keys
- SSE-C: client manages their own keys
    - Encryption is done in S3, but key is not stored in S3, you need encryption in transit using HTTPS
    - Encryption key must be provided in HTTP headers for every HTTP request made

- Client side encryption
    - You encrypt objects before uploading to S3, and decrypt when retrieving from S3
    - Client manages the keys and encryption cycle completely

2.3 DynamoDB

- DynamoDB Lab

NoSQL database with low read latency and can sustain storing hundreds of TBs.

- provisioned capacity: needs to specify rcu and wcu. if you go over, you can access a temporary burst capacity, but if you exceed that, you will get "ProvisionedThroughputExceededException"
    - Burst capacity only lasts for 300s
    - This error can come from exceeding provisioned rcu and wcu, or because of hot keys (one partition key being read too many times, for popular items), hot partitions, or very large items
    - To fix: exponential backoff, distribute partition keys and if rcu is an issue, use DAX
- on-demand: automatic scaling, no capacity planning needed
- you can switch between the modes every 24h

Each partition key is limited to 10GB of data, 3k RCU and 1k WCU

- 1 wcu = 1 write/s for an item up to 1kb. for larger files, more wcu
    - ex: write 10 items/s with item size 2kb: 10 * (2kb/1kb) = 20wcu
    - ex: write 6 items/s with item size 4.5kb: ~~6 * (4.5/1) = 27wcu~~ 4.5kb gets rounded to 5, so 6*5 = 30wcu
    - ex: write 120items/min with item size 2kb: 120/60 = 2items/s * 2 = 4wcu

for reading, 2 options: strongly(2x rcu) and eventually consistency. For consistent, for every API call, we set the ConsistentRead parameter

- 1 rcu = 1 strongly consistent reads/s = 2 eventually consistent read/s for an item up to 4kb
    - ex: 10 strong reads/s, item size 4kb: 10rcu
    - ex: 16 eventual reads/s, item size 12kb: 16/2 * 12/4 = 8*3 = 24rcu
    - ex: 10 strong reads/s, item size 6kb: ~~10 * 6/4 = 10 * 0.66 = 10rcu~~ 6 gets rounded to next multiplier of 4, which is 8, so 10 * 8/4 = 20rcu

> Read operations

- GetItem returns eventual reads by default, you can use projection expression to retrieve only certain attributes
- Query returns items based on a specific partition key
    - KeyConditionExpression, partition key and sort key
    - FilterExpression: additional filtering after the query operation (before data is returned to you). use only with non-key attributes (no hash or range attributes)
    - Returns: number of items specified on limit, or up to 1MB of data. Can paginate results
    - What can get queried? A table, a LSI or GSI
- Scan
    - Scan entire table and then filter out
    - Returns up to 1MB of data, use pagination to keep reading
    - Consumes a lot of RCU

    - For faster performance, use Parallel scan

Batch operations allow to save in latency by reducing number of API calls, operations done in parallel.

- BatchWriteItem:
    - up to 25 PutItem/DeleteItem per call
    - up to 16MB of data written
    - up to 400KB data per item
    - But not useful for update items, for that do it individually
- BatchGetItem
    - Returns items from one table or from more
    - Up to 100 items
    - Up to 16MB of data
    - Items are retrieved in parallel

A transaction is an operation to update multiple values at the same time. If one should fail, all should fail. All or nothing.

> LSI

- Alternative sort key, but must keep the same partition key as that of base table -> PK must be composite
- Up to 5 LSI per table
- Must be defined on table creation time

> GSI

- Alternative primary key from the base table (hash or hash+range)
    - PK can be simple or composite
- Used to speed up queries on non-key attributes
- Can be added/modified after table creation
- Must provision rcu and wcu for each index

If writes are throttled in GSI, the main table will be throttled as well

If we have a table with Users, their comments and the date. Each comment has an ID, which is the primary key. sort key is the date. if we want to find comments made by a user, we can either scan the whole table, or create a gsi with the username as partition key and date as sort key. after we have that, we can query on the gsi, by specifying the gsi name on the --index-name.

> DAX

- Fully managed, highly available in-memory cache for DynamoDB
- Microsecond latency for reads and queries
- solves the hot key problem
- 5min TTL for cache by default
- up to 10 nodes per cluster
- 3 nodes minimum in production, multi-az

dax vs. elasticache

- dax has individual objects cache for querys and scans
- elasticache stores aggregation results

DynamoDB Streams can deliver to Lambda, KCL. Not to Firehose!!

> Backup

- on demand: backups remain forever, even after table is deleted
    - create full backups of your tables for long-term retention and archival for regulatory compliance needs
- point in time recovery
    - helps to protect your DynamoDB tables from accidental write or delete operations
    - you can restore that table to any point in time during the last 35 days
    - rolling windows way, backup stays for 35 days
    - You can restore the table to the same AWS Region or to a different Region from where the backup resides. You can also exclude secondary indexes from being created on the new restored table. In addition, you can specify a different encryption mode.

if you want more control over backups, aws backups offers centralization, data protection and backup management, monitoring status of backups, verifying compliance. with aws backup we can create daily backups and store them for a month.

- Backup vault: a container that you organize your backups in. when creating a vault, a kms key is automatically created
- Backup plan: a policy expression that defines when and how you want to back up your AWS resources. The backup plan is attached to a backup vault.
- Resource assignment: defines which resources should be backed up. You can select resources by tags or by resource ARN.
- Recovery point: a snapshot/backup of a resource backed up by AWS Backup. Each recovery point can be restored with AWS Backup.

3. Processing

3.1 Glue

- AWS documentation
- Glue studio workshop
- Glue workshop

Glue is a serverless service that discovers and connects to diverse data sources (S3 data lakes, RDS, Redshift, most other SQL databases, DynamoDB) and manage your data in a centralized *data catalog* or Hive metastore, which is a persistent metadata store composed of many tables. You can visually create, run, and monitor ETL pipelines to load data into your data lakes. Also, you can immediately search and query cataloged data using Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum.

Hive is a service for running sql-like queries from EMR. the Data catalog can serve as a Hive metastore, and also, you can import a hive metastore into glue.

Only 1 data catalog per region allowed.

Glue crawler

Glue crawler is a program that populate the AWS Glue Data Catalog or Hive metastore with tables (cannot be used for DynamoDB, RDS). It connects to a data store, progresses through a prioritized list of classifiers to determine the schema for the data, and then creates metadata tables or updates existing tables in glue data catalog. Once catalogued, you can treat your unstructured data as if it was structured. The data is still in the data store, Glue only stores the table definition.

A classifier reads the data in a data store, if it recognizes the format of the data, it generates a schema. classifier also returns a certainty number to indicate how certain the format recognition was. glue provides built-in classifiers, but you can create custom ones too and if yo do, they are invoked first. if a classifier returns certainty=1.0, then it's 100% certain that it can create the correct schema. if no classifier returns certainty=1.0, glue uses the output of the classifier with the highest certainty, and if no classifier returns a certainty higher than 0.0, glue returns the default classification string of UNKNOWN.

Glue crawler extracts partitions based on how the S3 data is organized. e.g. if you have sensor data from devices every hour, you can a) query primarily by time range? then organize your buckets as yyyy/mm/dd/device. b) if you can query primarily by device, then organize bucket as device/yyy/... If you will query data daily, then partition by date and sort by device key. Folders where data is stored on S3 (sales/year=2019/month=1/day=1) are mapped to partitions, i.e. columns in the glue table

To do modifications on the data catalog, only works if data catalog is in s3, in format json/csv/parquet/avro. if parquet, special code required. and nested schemas are not supported

- To update table schema, a) re-run the crawler or use enableUpdateCatalog/updateBehavior from the script
- To add new partitions, a) re-run the crawler or use enableUpdateCatalog and partitionKeys options
- To create new tables, enableUpdateCatalog/updateBehavior with setCatalogInfo enabled

For tables that should support custom naming, create a new table manually.

Glue can crawl data in different regions, specific paths in your account and also from another account.

Crawler might take too long because the data is frequently getting increased, or data is compressed (but if files are in Parquet, ORC or Avro format, decompressing won't help). To improve performance, combine smaller files to create larger ones, or run multiple crawlers.

Incremental crawl would work once you alredy have a data catalog, and you're already adding data. To create a data catalog for the first time, this won't work.

If there is a folder in S3 with many files, and 80% of files have schema1 and 20% have schema2, only one table is created with columns of both schemas. Threshold is 70%-30%. If the distribution is like this or more equal, then two tables are created, each having columns of one schema respectively.

Glue ETL

Glue ETL is a serverless simple ETL service that runs on Spark to compress, partition, convert schema, clean, enrich, find record matches, make complex transformations, before the data arrives in the target, which can be S3, RDS, Redshift or Glue Data Catalog.

It generates code in Python or Scala, and this code can be modified. Your own code in Spark or PySpark scripts can be imported. Only works with Spark, for other frameworks, use EMR. Does not support Hive.

ETL jobs can be event-driven, on demand, on schedule (min every 5mins).

Glue ETL works with DPUs (data processing unit), which can be up and downscaled. With job metrics, you can know the max. capacity in DPUs and change it to a higher number if you want better performance on the ETL job.

When migrating a large JDBC table, the ETL job might run for a long time and eventually fail because of disk space issues (lost nodes). To resolve this issue, read the JDBC table in parallel.

Job bookmark persist state from the job run and prevents reprocessing of old data. For data that comes frecuently, use this to only process new data, not all data. Works with S3 and relational db-s via JDBC if primary keys are in sequential order, and only if data is coming as new rows, not as updating rows. If Job bookmarks are enabled but still there is duplicate data, check that object.commit() is present and that if you have multiple concurrent jobs, set max concurrency to 1.

ETL capabilities:

- DropFields, DropNullFields to drop null fields, Filter (filter records, extract one part of data), Join, Map (add, delete fields),
- FindMatches: identify duplicate or matching records in your dataset, even when the records don't have a common identifier and no fields match exactly
- Automatic format conversions: csv, json, avro, parquet, orc, xml
- Anything Spark can do, Glue ETL can do, e.g. k-means
- ResolveChoice: deals with ambiguities in a DynamicFrame and returns a new one. for example, two fields in the DF with the same name

Glue ETL supports serverless streaming ETL, it can consume from kinesis or kafka, clean and transform in-flight, and store results in S3 or elsewhere. It runs on spark structured streaming.

> Visual UIs

Glue studio is a visual interface for ETL workflows. you can create DAGs for complex workflows to transform/sample/join data, to consume from s3/kinesis/kafka/jdbc, and target to s3/glue DC. Visual job dashboard shows overview, status, run times.

Glue databrew is a visual data preparation tool for pre-processing large datasets. It's more specific than glue studio but for simple workflows take a data source, apply a transformation from the list of 250 ready-made transformations, and put output in S3. It can integrate with KMS (customer master keys only).

> Costs

- Crawler: billed by the second
- ETL jobs: billed by the second
- Glue data catalog: first 1M objects stored and accessed are free
- development endpoints for developing ETL code, like notebooks: charged by the minute

With Glue workflow, you can connect ETL activities like crawlers, jobs. You can use triggers to execute a job to run after a crawler is completed. so you would have one trigger to trigger the crawler, and another trigger to

trigger the job after the crawler is complete.

> Access

- Resource-based policies: policies attached to Glue resources, used to give IAM users and roles access to metadata definitions of databases, tables. Not to give or deny access of the Glue Data Catalog to one or other region. Follows best practice for least-privilege, e.g. to give permission to access a specific table
- Identity-based policies: policies attached to IAM users, to grant them permissions to create, access, or modify an AWS Glue resource

3.2 Lake formation

High-level service using Glue (to know the schema) and S3 to create **secure** data lakes from S3, on-prem. Loads data and monitors data flows, sets up partitions, helps with encryption and managing keys, defines transformation jobs and monitors them. Supports cross-account access.

If data in S3 is already in *Parquet* format, no need to use blueprint.

Steps in lake formation:

1. Create a data lake administrator
2. Register a **S3 path**
3. Create a database
4. Grant permissions
5. *Crawl* the data with AWS Glue to create the metadata and table
6. Grant access to the table data
7. Query the data using Amazon Athena
8. Add a new user with restricted access and verify the results

> Security

- Supports table-level and column-level data protection
- For cross-account datalakes, recipients must be set up as datalake administrator
- You can use aws RAM (resource access manager) for accounts external to your organization
- LK doesn't support manifests in Athena or Redshift queries
- To encrypt data catalogs in LK, you need IAM permissions on the KMS encryption key
- "Governed Tables" support ACID transactions across multiple tables. You can set up granular access control with row and cell-level security

Billing: Lake formation doesn't cost anything, but the underlying services do: Glue, S3, etc.

3.3 EMR

- EMR developer experience workshop

PaaS service for Elastic MapReduce on EC2 instances for ETL. If one node fails, provisions new nodes. When creating a cluster, specify framework (Spark, Hive...) and apps. EMR is charged by the hour plus for the additional EC2 charges.

- Master node: leader node, manages cluster

    - Tracks status of tasks, monitors cluster health
    - One single EC2 instance
    - Recommended instance type: m5.xlarge if there are less than 50 nodes
    - High availability support by having multiple master nodes in 1AZ, not in several AZs. EMR cluster can only be in one AZ
- Core node: runs tasks and stores data in HDFS
    - Can be scaled up and down to increase processing and HDFS capacity
    - Multi-node clusters have at least one core node
- Task node: runs tasks, does not store data
    - Optional node in the cluster
    - No risk of data loss when removing
    - Suitable for spot instances
        - If the app is SLA bound and the task node should complete even at on-demand price, then choose "switch to on-demand"

Cluster types:

- Transient cluster
    - Load data, process, store, then shut down. saves money
- Long-running clusters
    - Must be manually terminated. termination protection is on by default
    - Cluster becomes a data warehouse with periodic processing on large datasets
    - Can spin up task nodes using spot instances for temporary capacity
    - Can use reserved instances on long-running clusters to save money

To run scripts immediately after creating cluster:

- Upload the required scripts in Amazon S3 and execute them using custom bootstrap actions
- Provision an EC2 instance with Linux and run the scripts on the instance. Create an AMI using this instance and then use this AMI to launch the EMR cluster

EC2 instance types, you can only choose one for all nodes in the cluster

- Instance groups
    - Only one AZ
    - Only one instance type per node type: master, core, task
    - autoscaling provided
- Instance fleet
    - choose AZ based on subnet
    - multiple instance types per node type
    - no autoscaling

> Storage options in EMR

- Ephemeral data storage
    - Local filesystem for buffers, caches
    - HDFS (Hadoop distributed file system): distributes data blocks across cluster for redundancy. Useful for caching intermediate results with significant random I/O

    - EBS for HDFS: EBS volumes can only be attached when launching a cluster. If you manually detach an EBS volume, EMR treats it as a failure and replaces it
- Persistent, EMRFS: it's an object store in S3
    - Access S3 as if it were HDFS. If high network costs, add prelimiary step that will use S3DistCp
    - Can't be used for root volume, or Hadoop storage layer

Scaling

Scaling supported for spot instances, on-demand and those instances in a savings plan. Scaling is available only for Spark, Hive and Yarn (Yet Another Resource Negotiator: manages cluster resources for data processing frameworks)

- Scale-up strategy: first add core nodes, then task nodes, up to max units specified
- Scale-down strategy: first remove task nodes, then core nodes, no further than min constraints
- Spot nodes are always removed before on-demand nodes

Security

- For authentication, Kerberos
- EC2 security groups: one for master node, another for cluster node. Security groups allow node-to-node communication
    - To get access to the cluster, add your IP as a TCP inbound rule in the cluster's security group
- Don't grant full access to S3 for the EMR cluster's service role. Instead, attach IAM roles to EMRFS and limit the EMRFS and S3 permissions attached to the service role for the cluster EC2 instances

Encryption

- EMRFS encryption
    - S3 encryption at rest: sse-s3 (only AES-256 supports audit logging), sse-kms, client-side. ss3-c not supported
    - With S3 encryption in EMR, all encryption modes use a single CME by default to encrypt objects. To access encrypted data in S3, modify the cluster's security configuration by delegating the appropriate CKS for each bucket under the per-bucket encryption overrides
    - EC2 instance store encryption, NVMe or LUKS encryption
    - EBS volumes: KMS (works with root volume), LUKS (doesn't work with root module)
    - TLS in transit between EMR nodes and S3

For log file encryption:

- Enable logging when creating EMR cluster
- By default, each Amazon EMR cluster writes log files on the master node. These are written to the /mnt/var/log/ directory
- With Amazon EMR version 5.30.0 and later (except Amazon EMR 6.0.0), you can encrypt log files stored in Amazon S3 with an AWS KMS customer-managed key

EMR serverless

Automatic provisioning of workers, amount and scheduling for jobs. You can specify default worker sizes and pre-initialized capacity. Only one region supported. Choose EMR release and runtime (spark, hive, presto) and submit queries/scripts via job run requests.

To use EMR serverless:

1. IAM user
2. Use aws cli to set up job execution role, allow emr-serverless service, s3 access, glue access, kms keys
3. Create EMR serverless app
4. Add job (e.g. spark script, hive query) within this app
5. Obtain outputs and logs

Spark

Distributed processing framework for big data analysis (OLAP). Allows more depth into Spark than Glue (which is serverless)

- Has in-memory caching, optimized query execution
- Supports Java, Scala, Python and R
- Supports code reuse across
    - Batch processing
    - Interactive queries (sparksql)
    - Real-time analytics
    - ML (MLlib)
    - Graph processing
- Spark streaming can deliver to KDS, consume from KDS and Kafka
- Integration with Redshift to do ETL on Redshift directly
    - Usecase: lots of data in S3, open to Redshift, start a spark cluster on EMR, use that to do ETL and write again to Redshift.

Spark apps are run as independent processes on a cluster. SparkContext (driver program) coordinates them and uses Cluster manager. It sends app code and tasks to executors, which run computations and store data. To run the spark script, do `spark-submit my_script.py`

Apache Pig

Service to accelerate writing mappers and reducers, with Pig Latin, a scripting language with SQL-like syntax. Highly extensible with user defined functions.

Apache HBase

NoSQL petabyte-scale database, based on Google's BigTable, on top of HDFS. It's very fast because it does processing in-memory. Hive integration. It's very similar to DynamoDB. HBase advantages:

- Efficient storage of sparse data
- Appropriate for high frequency consistent reads and writes
- High write and update throughput
- More integration with Hadoop

DynamoDB:

- Fully managed (auto-scaling)
- More integration with other AWS services, like Glue

HBase can store following data in S3: snapshots, HBase files and metadata, read-replicas.

Use EMR File System (EMRFS) to store the data and enable the EMRFS consistent view feature. Launch two separate EMR clusters in two different Availability Zones. Launch a primary and secondary EMR HBase cluster. Configure multiple master nodes in the primary cluster and create a secondary read replica cluster in a different Availability Zone. Point the two clusters to the same S3 bucket and hbase.rootdir location.

> Hive

Interactive SQL on unstructured data in EMR. It uses HiveQL, similar to SQL. It's the most appropriate way for data warehouse application, more comfortable than Spark. Integration with S3 and DynamoDB. Allows custom code.

By default, Hive has a metastore info in a MySQL on the master node's filesystem. External metastores like Glue Data Catalog, RDS or Aurora offer persistence, centralization so that many users can access, and better resiliency if the cluster shuts down.

MSCK REPAIR TABLE command scans a file system such as Amazon S3 for Hive compatible partitions that were added to the file system after the table was created. It compares the partitions in the table metadata and the partitions in S3. If new partitions are present in the S3 location that you specified when you created the table, it adds those partitions to the metadata and to the Athena table.

> Presto

Interactive queries at petabyte scale for OLAP. It can connect to many different big data databases and data stores (Redshift, RDS) at once, and query across them. Athena is basically serverless Presto. Custom code not allowed,´.

- EMR notebook
  - many AWS integrations: backup in S3, hosted in VPC
  - access only via AWS console
  - use your own code to interact with cluster, e.g provision cluster
- Apache Zeppelin
  - notebook to run Python on EMR data
  - Notebook notes, share with others
  - Integration with Spark
  - Execute SQL directly against SparkSQL
  - Chart and graph visualization

- Hue (Hadoop User Experience): graphical frontend for the cluster
- Splunk: operational tool, can be used to visualize EMR and S3 data using the EMR hadoop cluster. Makes machine data accessible, usable and valuable
- Flume: for real-time streaming applications
- Sqoop: to transfer data between Hadoop and relational databases
- MXNet: deep learning on EMR
- S3DistCP: to copy large amounts of data S3 <-> HDFS, or between S3 buckets. Doesn't support concatenation of Parquet files, so use PySpark for this
- Ganglia: to monitor cluster's performance as a whole and with individual nodes

3.4 Data pipeline

Managed ETL service for scheduling regular data movement and processing, running on EC2 (which automatically has Task runners installed and executed)

Define the business logic of your data management in the pipeline definition, then DP determines the tasks, schedules them and assigns them to task runners. You can write a custom task runner application, or you can use the Task Runner application that is provided by Data Pipeline.

The pipeline is a container that has:

- Data nodes: destination of data
- Activities, like copying
- Pre-conditions, like readiness check
- Schedules

Features:

- Runs on EC2 or EMR clusters
- Provisioning and termination is automatic, depending on the tasks
- Tasks can depend on other tasks, they can wait for the previous tasks
- Supports for task retries, and re-assignation to another task runner
- Integrated with on-prem and cloud
- Integration with S3, RDS, DynamoDB and Redshift for storage
- Supports cross-region pipelines
- Supports pre-built templates

Example use case: copy log files from EC2 to S3 daily. Launch data analysis from these S3 files into EMR weekly.

4. Analysis

4.1 Kinesis Data Analytics

Serverless, real-tme analytics service, but not cheap. Can be used for ETL, schema discovery, metric generation, analytics. It can deliver to KDS, KDF, Lambda, which these can write to S3 or DynamoDB.

KDA uses Kinesis Processing Units (KPU). A single KPU provides 4 GB of memory and corresponding computing and networking. The default limit for KPUs for your application is 8 KPUs, so the default capacity of the processing app in terms of memory is 32GB.

With RANDOM_CUT_FOREST, a SQL function, KDA can detect anomalies on numeric columns in data stream.

The input to KDA comes as unbounded data flowing continuously, and to get results sets, queries are bound using a window defined in terms of time or rows.

- Sliding windows: aggregates data continuously
    - Fixed time or row-count interval windows
- Stagger windows: aggregates data using keyed windows that open as data arrives
    - Time-based windows
    - Allow for multiple overlapping windows
    - Good for data that arrives at inconsistent times
    - They reduce late or out-of-order data

- Tumbling windows: aggregates data using distinct windows that open and close at regular intervals
    - Time-based windows
    - Individual records might fall into separate windows, so partial results must be combined later
    - Don't reduce late or out-of-order data

KDA can optionally use reference tables to enrich data coming from streaming. This mapping data must be stored in S3. When the app starts, KDA reads the object and creates an in-replication reference table. You can write a SQL query to join streaming data with reference data. To refresh the data after the reference table has been created, explicitly call the UpdateApplication API.

Lambda can also enrich logs.

> Apache Flink

Apache Flink is a framework for processing data streams, can work with Java and Scala. KDA integrates Flink with AWS, making it serverless. Instead of using SQL, you can develop your own Flink app from scratch and load it into KDA via S3. Sources for Flink can be KDS, or MSK.

4.2 Opensearch

Fully managed petabyte-scale analysis (OLAP) and reporting service with *visualization*, originally started as search engine. Very storage heavy.

Accepts only JSON format, not CSV.

Main concepts:

- Not suitable for ad-hoc data querying, for that use Athena
- Cheaper alternative than KDA for log analysis
- Data import allowed from: Kinesis, DynamoDB, Logstash / Beats, and Elasticsearch's native API
- Snapshots stored in S3

When creating cluster, you have to decide:

- Number of nodes in cluster and instance types
    - Recommended: 3 dedicated master nodes, 3 data nodes
    - With two, you can have split-brain problem where we don't know which of the two nodes is the actual primary node. With three, the third master node decides which would be the primary
- Number of shards
- Storage requirement (otherwise, out of disk space error risk)
- Autoscaling possibility
- Whether cluster is in a VPC
    - How to access cluster if it's in VPC
        - Cognito
        - Nginx reverse proxy on EC2 forwarding to ES domain
        - SSH tunnel for port 5601
        - VPC direct connect
        - VPN

Pricing: Pay for what you use, instance-hours (charged even if idle: if not using them, just shut them down), storage, data transfer.

Search concepts

- Documents: things you're looking for, can be text, json. Every document has a unique ID and a type
- Types: they define the scema and mapping shared by documents that represent the same sort of thing. Still in ES6, but obsolete in future versions
- Indices: an index powers search into all docs within a collection of ypes. They contain inverted indices that let you search across everything within them at once

An index is split into shard, each shard might be on a different node in a cluster. Every shard is a self-contained mini search engine. The index has two primary shards and two replicas.

- Write requests are routed to the primary shard, then replicated
- Read requests are routed to the primary or any replica

Index state management (ISM) automates index management policies. They run every 30-48 mins, random jitter to ensure they don't all run at once. It can send notification when it's done

- deletes old indices after a period of time
- moves indices into read only state after some time
- move indices between storage types
- roll up old data into summarized indices, in case you don't need the whole old data but just the summary. saves storage costs
- reduce replica count over time
- automate index snapshot

To ensure high availability or replicate data geographically for better latency, you can replicate indices/mappings/metadata across domains (clusters). The "follower" index (replica) pulls data from "leader" index. to enable this, we need fine-grained access control and node-to-node encryption.

Remote reindex allows copying indices (not the whole cluster) from one cluster to another on demand.

Supports dashboard with Kibana, which can auto-refresh when data changes.

Storage types

Data can be migrated between different storage types

- Hot: used by standard data nodes, instance stores or EBS volumes. fastest performance, most expensive
- UltraWarm: uses S3 and caching, best for indices with few writes (like log data or immutable data), slower performance but much lower cost. Requires you to have a dedicated master node
- Cold: uses S3, suitable for periodic research or forensic analysis on older data, must have dedicated master and have UltraWarm enabled, not compatible with T2 or T3 instance types on data nodes

Errors

JVMMemoryPressure error: reduce amount of shards by deleting old or unused indices, having too many small shards can be bad. shards should be small enough that ES can handle them but not too small that they place needless strain on the hardware

4.3 Athena

- User guide

Serverless service for interactive/ad-hoc simple SQL queries from data sources, but data stays in its original location. It uses Presto under the hood.

For each dataset that you'd like to query, Athena must have an underlying table it will use for obtaining and returning query results. Therefore, before querying data, a table must be registered in Athena, automatically or manually. The metadata in the table shows data location in Amazon S3, and specifies the structure of the data, for example, column names, data types, and the name of the table.

How to optimize performance:

- Convert files to columnar formats
- Compress
- Have few big files, instead of many small files
- Use partitions
- Data in the same region
- No need to randomize prefix naming for the key

Good solution for:

- ad-hoc queries of logs for example
- querying staging data before loading to redshift
- log analysis
- Integration with Jupyter, Zeppelin, Rstudio notebooks
- Integration with QuickSight
- Integration via ODBC/JDBC with other visualization tools

In CTAS (CREATE TABLE AS SELECT) queries,

- Partitioning CTAS query results works well when the number of partitions you plan to have is limited and the partitioned columns have low cardinality
- Bucketing CTAS query results works well when you bucket data by the column that has high cardinality and evenly distributed values

S3 prefix configuration (Athena might return empty after a query):

- Athena doesn't support table location paths that include double slash (//)
- Athena doesn't support table location paths that start with an underscore (_)
- If a table has been created for each file stored under the same prefix, create one prefix per table

To make a Hive-metastore compatible solution: use a key prefix of the form **date=**year-month-day/ for partitioning data and use MSCK REPAIR TABLE statement to load the partitions.

If table appears in Glue but not in Athena:

- Data might have come to Glue from other data sources. Athena only shows data from S3 (but not Glacier!)
- Tables can have formats that aren't supported by Athena, such as XML (but JSON is supported)

User configuration

- A Workgroup can be composed of users/teams/apps/workloads
- Workgroups allow isolation, query access control, query metrics for each query, cost tracking, cost constraint enforcing
- IAM groups cannot manage the isolation of queries and tracking query history for groups

Limits:

Two types of cost controls:

- per-query control limit: specifies the total amount of data scanned per query. If any query that runs in the workgroup exceeds the limit, it is canceled. You can create only one per-query control limit in a workgroup and it applies to each query that runs in it
- per-workgroup control limit: you can set several limits per Workgroup, e.g. hourly, daily aggregates on scanned data. Specifies the total amount of data scanned for all queries that run in the entire workgroup, not on a specific query only

Pricing

- Charged by TB scanned
- Charged only for successful or cancelled queries, not for failed queries
- Using columnar formats improves performance and also saves costs
- Two types of cost controls: per-query limit and per-workgroup limit

Athena provides ACID transactions, concurrent users can safely make row-level modifications. You can also recover recently deleted data with SELECT statement.

4.4 Redshift

Fully-managed, petabyte-scale data warehouse for structured data (for unstructured, use ETL on EMR). Supports complex analytical queries with SQL and joins (OLAP).

Features:

- On-demand scaling, vertical and horizontal, with downtime
    - When upscaling, a new cluster is created and the old one remains available as read. then the CNAME flips to new cluster (downtime), then data is moved in parallel to the new cluster
- Automatic replication to another region but limited to one AZ
    - To maintain a read-replica in another AZ, spin up separate clusters in 1+AZs using Kinesis to simultaneously write data into each cluster. Use Route53 to direct your analytic tools to the nearest cluster when querying your data
- Snapshots and backed up to S3
    - Automated: snapshot every 8h or after 5GB/node of data changes
    - Manual: taken at any time, default retention is indefinite even after cluster deletion
- Datalake can be exported to S3 in Parquet format
- Not low-latency
- Parallel processing, columnar data storage and columnar compression

Cluster features:

- Cluster is composed of leader node, connecting to the client via JDBC/ODBC, and the compute nodes
- Automatic data replication within the cluster
- Failed nodes or drivers are automatically replaced
- Node types:
    - Dense storage nodes: just storage, no compute
    - Dense compute nodes: for queries
- Sort keys determine the order in which rows are stored in a table

Other:

- Redshift workload management (RWM): tool to prioritize short, fast queries vs. long, slow queries. You can also create query queues. When running a query, WLM assigns a query to a queue according to the user group
- Short query acceleration (SQA) uses ML to predict the execution times of queries in advance, and then routes short queries to a dedicated queue. Use when you only have short queries, if you also have long queries, use RWM

Data distribution

When creating table, you have to choose distribution styles. These decide how data is distributed among the nodes and the slices. Best practices: data should be distributed in such a way that the rows that participate in joins are already collocated on the nodes with their joining rows in other tables. The styles:

- DISTSTYLE AUTO: the service decides based on data size, combination of ALL and EVEN. Cannot do KEY
- DISTSTYLE ALL: entire table is copied to every node
    - good for static or tables updated infrequently that are used in joins or as lookup tables
    - fast queries, but takes a lot of space. And actions on this table like loading, deleting, takes long
    - don't use it for: small tables (<500KB), tables that will be used once, tables updated frequentiy, or tables with >10M rows
- DISTSTYLE KEY: rows distributed based on the values in one column, leader node places matching values on the same node slice
    - good for tables where there are many rows with a single identifier that **will be used for queries and joins**, like the main table of a service
- DISTSTYLE EVEN: rows distributed across slices in round-robin
    - good for tables that don't participate in joins
    - good for tables with many unique values
    - good for tables that change frequently
    - good for tables connected to visualization tools

IF two tables have the column that will be queried upon / joined, use DISTSTYLE KEY for both, and use this column as the DISTKEY for both tables.

Best practices:

- Designate a common column for fact table and dimension table
- Select DISTKEY with high cardinality
- Choose the largest dimension based on the size of the *filtered* dataset

Scaling

Concurrency scaling, when there is high demand, adds query processing power to a cluster to specific users or queues as needed. Second-latency, provides users with fast, consistent performance even when there are a lot of queries. It uses ML to predict when queueing might start to happen. Not every query needs a concurrency scaling cluster, only those with high priority. Concurrency scaling is not a substitute for good table and query design.

Classic resize	**Elastic resize (recommended by AWS whenever possible)**
Change node type / number of nodes / both	Change node type / number of nodes / both
Can operate on single-node cluster	Can't operate on single-node cluster
Good for making permanent changes to cluster (double, halve)	Good for temporary spikes in demand
Creates new cluster	Creates new cluster only if node type is changing
Retains sort order, not system tables and records marked for deletion	Retains system log tables, not disk size
Downtime for hours/days -> read-only	Downtime of 10-15mins -> read-only

To create external tables, define the structure for files and register them as tables in glue data catalog.

> Data import / export

Data can be loadad into Redshift from S3, EMR, DynamoDB, EC2 or remote hosts

- COPY: parallel command to load files into Redshift. The table must exist in Redshift, otherwise error
  - To optimize data import:
    - File sizes between 1MB and 1GB, ideally 1–125 MB after compression: if you have smaller files, unify them. if you have bigger files, split them
    - All files similar size
    - Compress files
    - Number of files should be a multiple of slices in the cluster
  - This command adds hidden metadata columns
  - This command can also decrypt data
  - For narrow tables (many rows, few columns), load it with one COPY transaction
  - Manifest file:
    - good option if dataset spans many files, or different prefixes, different buckets
    - this file ensures that only the correct data is loaded, loaded only once, and provides accountability. then use copy with this manifest file
    - Avoids the problem of Resume Building Event
- INSERT: move data from one table to another, data is already in Redshift
- VACUUM: recover space from deleted rows, it re-sorts rows and reclaims space. Can only be done by the table ownser or superuser
  - It can be paused if there is high load in the cluster
  - When deleting rows, they are not automatically deleted, but rather marked for deletion. When updating rows, new row is appended and old row is marked for deletion

    - If VACUUM is slow
        - Unsorted data
        - More columns on the table
        - VACUUM is run too infrequently
    - check VACUUM status by using the svv_vacuum_progress query
- UNLOAD: Glue unloads from a Redshift table into S3
- DBLINK: copy and sync data between PostgreSQL and Redshift

To avoid deduplication when loading data into Redshift:

1. Create a temporary staging table configured like the destination table in Redshift
2. COPY records to this staging table
3. JOIN staging table with target table
4. UPSERT: merges data by updating existing records and adding new data
5. Delete staging table

If you use UPSERT to COPY new data into a table, the table needs to be sorted

Queries

Redshift Spectrum

- Serverless service on top of Redshift (so you have to provision infra) to query exabytes of unstructured data in S3 without loading
- Can create tables (CREATE EXTERNAL TABLE) in Redshift pointing to S3
- Limitless concurrency, horizontal scaling
- Glue data catalog + Athena has console-based query sql engine, with Spectrum it's like a table in the Redshift database
- Redshift used for reporting and analysis, Glue for complex transformations
- Athena is faster and cheaper, just runs queries against S3 data

AQUA: accelerated query accelerator, for apps that use S3 and Redshift. Pushes reduction and aggregation queries closer to the data, it has high-bandwidth connection to S3.

Redshift serverless

- Automatically scale and provision for your workload
- Uses ML to maintain performance across variable and sporadic workloads
- Spectrum, public endpoints not supported

Security

Enhanced VPC routing: forces Redshift to use the VPC for all COPY and UNLOAD commands, and therefore traffic appears on the VPC Flow logs.

HSM = Hardware security module

- Encryption at rest using KMS or an HSM device, needs establishing a connection
    - When using HSM, you need to create a new HSM-encrypted cluster, migrate to this new one, and use client + server certificate to configure a trusted connection

    - If you migrate an unencrypted cluster to an HSM-encrypted cluster, you must create the new encrypted cluster and then move data to it
- Encryption in-flight using the JDBC driver enabled with SSL
- You can define access privileges for users or groups, with grant or revoke commands in sql
- To meet security and compliance requirements, set column-level grant and revoke

To copy a snapshot from a KMS-encrypted cluster in another region for backup, then:

1. Create a snapshot copy grant in *destination* region for a KMS key in the *destination* region
2. Configure Redshift cross-region snapshots in the *source* region

5. Quicksight

Serverless business intelligence visualization tool. Supports simple ad-hoc analysis and limited ETL. Used to create and share dashboards privately with specific users, not with the general public.

For public facing interactive charts and dashboards, write data into S3, enable Cloudfront, and use Highcharts or d3.js to visualize the data on the web.

Data sources:

- Redshift
    - Can connect to Redshift regardless if the cluster is in a VPC or not
    - If cluster is in another VPC/Region, create a new security group with an inbound rule authorizing access from the IP range of QuickSight servers in that region
    - If cluster is in another account, provide the source's connection details
- Aurora/RDS
- Athena
- EC2-hosted dbs
- Salesforce and other SaaS
- Files (on-prem, excel, s3, **but not in parquet format**)
    - If import into SPICE fails, edit S3 bucket permissions directly from Quicksight

> SPICE (Super-fast parallel in-memory calculation engine)

- Accelerates interactive queries on large datasets
- Each user gets 10GB of SPICE
- In direct query mode, using Athena, some queries might time out. Good use for SPICE. If it takes >30mins to import data from Athena into SPICE, that means big queries are timing out.

Datasets can be refreshed on a Daily, Weekly, or Monthly basis, but not faster. for that, use Kibana

> Visualization types

- AutoGraph: automatically selects the visualization
- Bar charts / histogram: for comparison and distribution
- Line graphs: for changes over time
- Combo chart: combination of two bar charts, two line graphs or bar chart + line graph
- Waterfall chart: to visualize a sequential summation as values are added or subtracted
- Funnel chart: to visualize data that moves across multiple stages in a linear process

- Scatter plots: for correlation
- Heat map: highlight outliers and trends
- Tree map: can be used to represent hierarchical data in the form of nested rectangles
- Pie/doughnut graphs, tree maps: for aggregation, to see how much percentage each section has, like a cake
- Pivot tables: for tabular data
- KPI: to visualize a comparison between a key value and its target value

> Enterprise

- Double cost as Standard, paid per user per month
- Extra Spice capacity
- Encryption at rest and in-transit with AWS-managed-key, can't import customer keys
- Row-level security enabled with dataset rules. You can explicitly allow or deny access based on the dataset rules. Allowing access is the default
    - Row-level security only works for fields containing textual data (string, char, varchar, and so on). It doesn't currently work for dates or numeric fields
- Microsoft AD integration
- Reader users (users that don't author anything, but just have read access. Can log in with QuickSight user/pass, SAML or AD)
- Supports 500M rows / 500GB dataset
    - Standard only 25M rows / 25GB dataset

> ML

- Random cut forest: find anomalies/outliers in the dataset, forecasting: detect seasonality and trends
- Autonarratives, adds "story of your data" to your dashboard
- Suggested insight: A new tab displays ready-to-use suggested insights
- Quicksight Q: allows queries with NLP, offered as add-on in some regions. To use this, you have to set up topics associated to datasets. Datasets and the fields must be NLP-friendly

6 Security

6.1 KMS

Manages and rotates keys, but only up to 4kb of data per call. If data is bigger, use envelope encryption. Supports symmetric and asymmetric encryption.

3 types of customer master keys (CMK)

- AWS managed CMK: free
- User keys created in KMS: $1/mo
- User keys imported: $1/mo

You also pay for API calls to KMS

S3 allows in-place encryption on the fly, other services like EBS, EFS, RDS, Elasticache, need snapshot/backup to be able to encrypt it.

Automatic key rotation is possible only for customer-managed CMK. If enabled, it rotates keys every year, old key is kept alive so you can decrypt old data. New key has the same CMK ID, only the backing key is changed.

You can also rotate keys manually, whenever you want. in this case new key has different CMK ID. Previous key is also kept alive. You should use alias to hide the change of key for the app.

The alias reduces the coding effort.

6.2 CloudHSM

AWS provisions the hardware (HSM = hardware security module), but you have to use your own client to perform the encryption. You manage your own encryption keys entirely. Supports both symmetric and asymmetric encryption, multiAZ can be enabled.

Redshift supports CloudHSM. CloudHSM is a good option for SSE-C encryption.

6.3 STS

Security token service, allows to grant limited and temporary access to AWS resources. Token is valid for up to 1h

Allows cross-account access:

1. Define IAM role for another account to access
2. Define which accounts can access this role
3. Use STS to retrieve credentials and assume the role that you have access to

Also supports federation, users outside of AWS get temporary role to access AWS resources: AD, SAML, SSO, Cognito