https://zlliu.medium.com/raw-practice-notes-dump-aws-data-engineer-associate-week-1-e2efbda5c6be

SQS Queue Stuff Events that can remove messages from SQS queue

DeleteMessage API call — typical method to remove messages from a queue maxReceivedCount has been reached for message — how many times message can be received before message is deleted and added to dead-letter queue queue is purged → remove all messages in queue without deleting queue (usually to reset an app) Standard SQS queue VS first in first out (FIFO) queue

standard → unlimited throughput — nearly unlimited number of API calls per second per API action FIFO → high throughput → supports up to 300 messages per second per API method. If batching, 3000 messages per second (300 batches of 10 messages) standard → at-least-once delivery (might be more than once) FIFO → exactly once processing (one and only time, in order) standard → best effort ordering → might be in slightly different order FIFO → first in first out delivery → order strictly preserved SQS visibility timeout meaning

when consumer A is processing message in queue, SQS temporarily hides this message from other consumers. by setting visibility timeout on message visibility timeout → time period where other consumers (not A) are prevented from receiving and processing this particular message default is 30 seconds, max is 12 hours eg. if app takes 10 seconds to process a message, but visibility timeout is 2 seconds, this message might be consumed by another consume, leading to duplicates Dead letter queues

contains messages that cant be processed successfully from other queues (used for debugging) DLQ for FIFO queue must also be FIFO Some Amazon SQS Actions/Attributes

DeleteMessage → delete specified message from specified queue (use receiptHandle instead of messageId) PurgeQueue → deletes available messages in queue (by QueueUrl) ReceiveMessage → retrieve up to 10 messages from specified queue SendMessage → deliver message to specified queue NOTE — ReceiveMessage API call does not delete message from queue. consumer A get data → visibility timeout invoked → other consumers cannot see this message for a period of time → consumer A finishes processing the message → consumer A tells queue to delete message using DeleteMessage

Metadata store w fine-grained permissions Centralized metadata storage solution (reliable, scalable) + ensure that fine-grained permissions can be controlled at database, table, col, row, cell levels

Use AWS Lake Formation to create data lake and data catalog + control access using Lake formation data filters Stuff that doesn’t work

AWS Glue → control permissions at table level with Glue Data Catalog, but cannot control at database, table, column, row, cell levels Aurora DB → also need to deploy Hive app (high operational overhead) Amazon EMR → cannot control access to Hive metastore AWS Glue Data Catalog with Amazon Athena

can only limit access to databases and tables. Fine-grain access controls apply at table level, cannot limit access to individual partitions Apache Hive on EMR

open source, distributed, provides data warehouse-like query capabilities enables users to read/write/manage lots of data using SQL-like syntax lots of data == petabyes (1000 terabyte == 1 petabyte) AWS Lake Formation → to create data lakes data lake == centralized repo to store structured/unstructured data at any scale (and any format) can manage fine-grained access control for data lake data on S3 + metadata in AWS glue data catalog fine-grained access controls goes to row, col, cell level even Redshift Stuff Redshift → petabyte-scale data warehouse service

Note: only pay when we query, no need pay if data is idle Data warehouse → giant database optimized for analytics

store large amounts of data (current & historical) from multiple sources ETL processes move data from other sources into data warehouse fixed relational schema, but sometimes semistructured is ok Redshift COPY Command

loads data into table from data files (eg S3, EMR) or DynamoDB ie other source data → redshift table Redshift JOIN command

like SQL → combine data from 2 or more tables Redshift Materialized View

contains preocmputed result set (based on SQL query etc) SELECT this materialized view → precomputed result returned without accessing base tables → faster useful for speeding up queries that are predictable and repeated Redshift UNLOAD command

unload result of query to TXT/JSON/CSV/PARQUET default is | delimited Cron stuff Cloudshell

browser-based, pre-authenticated shell we can launch directly from AWS management console. can run AWS CLI commands without having to install locally EventBridge Scheduler

serverless schedular allows us to create/run/manage tasks Cloud9

integrated development environment (IDE) usable through web browser Kinesis Data Streams with AWS Lambda To increase performance when reading kinesis data streams w lambda:

increase number of shards for kinesis data stream test different parallelization factor settings to find most performant register lambda function as consumer w enhanced fan-out Lambda default behaviour

lambda create 1 concurrent instance of lambda function for each shard 3 shards → 3 concurrent functions Kinesis Data Streams (KDS)

a kinesis data stream is a set of shards producer → sends data to kinesis data streams eg. log data consumer → get data from kinesis data streams shard → uniquely identified sequence of data records in a stream. Stream composed of 1+ shards. Each shard provides fixed unit of capacity eg. 5 transactions per second. data capacity of stream depends on number of shards WE specify partition key → used to group data by shard within stream sequence number → each data record has sequence number which is unique per partition-key within its shard server-side encryption → KDS auto encrypt data from producer as it enters a stream (KMS master keys for encryption) KDS — Shared throughput VS Dedicated throughput (enhanced fanout)

throughput → volume of data passing through network at given period shared throughput → no need propagation delays under 200ms dedicated throughput → consumers that use this dont need to share with other consumers enhanced fanout → consumers to receive records from stream with throughput of up to 2 MB of data per second per shard can register up to 20 consumers per stream to use enhanced fanout Lambda and KDS lambda polls each shard in kinesis stream using HTTP protocol shared throughput w other consumers of shard lambda reads records from data stream, invokes function synchronously (read in batches) to increase concurrency → process multiple batches in parellel (lambda can process up to 10 batches in each shard simultaneously) ParallelizationFactor → process ONE shard with MORE THAN ONE lambda function (shard from KDS or dynamoDB streams) eg. set ParallelizationFactor = 2 → 200 concurrent lambda functions at max to process 100 kinesis data shards → helps scale up processing througput when IteratorAge is high IteratorAge → age of last record in all GetRecord calls made in stream, regardless of which consumer called it high iteratorAge → data is not being processed in timely manner KDS Capacity modes

determines how data stream is managed + $$$ on-demand → no capacity planning, auto scale number of shards provisioned → WE specify number of shards unpredictable nature of data → on-demand more performant Lambda reserved concurrency

number of in-slight requests that function is currently handling reserved concurrency → max number of concurrent instances allocated to your function. no add $$$ provisioned concurrency → number of pre-initialized execution envs allocated to function, ready to respond to incoming requests. add $$$ Some Storage stuff Elastic Block Store (EBS)

scalable block storage that can be used with EC2 instances EBS volumes → we attach this to EC2 instance (kinda like hard drive) EBS snapshots → point-in-time backups of EBS volumes DynamoDB (serverless)

fully managed, noSQL db service + distributed database DynamoDB accelerator (DAX) → in memory cache Redshift Stuff Redshift → petabyte-scale data warehouse service

resources automatically provisioned, capacity auto scaled no $$$ when data is idle redshift query editor v2 OR other BI tool eg. quicksight cluster → one or more compute nodes leader node → manages comms with client programs + distributes SQL statements to compute nodes to get results Redshift Performance features

Massively parellel processing (MPP) → multiple compute nodes handle all query processing at once. Redshift distributes rows of table to compute nodes → faster Columnar data storage → reduce disk IO requirements + better perf Data compression → reduce storage requirements. when query, compressed data read into memory, uncompressed during query run Query optimizer → Result caching → caches results of certain query types in memory Redshift Materialized views

complex queries → eg multiple table joins and aggregations materialized view → precomputed result set (one or more tables) can SELECT a materialized view → results w/o touching base tables useful for speeding queries that are repeated and predictable Federated queries → can query across operational databases, warehouses, lakes + integrate queries with other external DBs

Redshift Data Sharing

securely share access to live data across redshift clusters data sharing write-access use case Redshift Manual Snapshots

ability to create point-in-time recovery backups (not cost effective) Redshift Serverless vs Provisioned

serverless → auto provisioned, auto scaled provisioned → self manage compute and leader nodes serverless → pay for workloads run provisioned → billing occurs per second when cluster isnt paused Redshift User Defined function (UDF)

Some Encryption Stuff AWS KMW → create/control encryption keys to protect data

Encrypting RDS

data encrypted at rest → storage, automated backups, read replicas, snapshots secure data from unauthorized access to underlying storage RDS auto integrate with KMS if want full control over KMS key → customer managed key AWS managed key → less hassle but less control Server side encryption w KMS (SSE-KMS)

all S3 buckets have encryption configured by default all new objects uploaded are auto encrypted at rest server side encryption w S3 managed keys (SSE-S3) is default encryption AWS Glue DataBrew

visual data preparation tool, gives let us vlean n normalize data without need to write code. provides data masking mechanisms 250 read-made transformations to automate data prep tasks eg filtering anomalies, converting data to standard formats, correcting invalid values → speed in data prep Identify n mask PII data in DataBrew Sagemaker Data Wrangler

end-to-end solution to import, prep, transform featurize analyze data import, data flow, transform, generate data insights, analyze, export if use with PII, this exposes PII to ML model AWS Glue Stuff serverless data integration service → discover, prep, move, integerate data from multiple sources + ETL Glue Studio → GUI to manage integration jobs Discover and organize data

unify and search across multiple data stores → store, index, search across multiple data sources by cataloging all data auto discover data → AWS glue crawlers to infer schema info, integrate into AWS glue data catalog mange schemas n permissions → control access to db and tables connect to many data stores → glue connections to build data lake Transform, prep, clean data for analysis

drag n drop ETL interface complex ETL pipelines + simple job scheduling → invoke aws glue jobs based on schedule, on demand or based on event clean n transform streaming data in transit deduplicate and cleanse data w built-in ML built-in job notebooks → AWS Glue Job Notebooks AWS Glue sensitive data detections → find PII stuff in process Build nad monitor data pipelines

auto scale based on workload automate jobs w event-based triggers run AWS glue jobs with eg, spark or Ray. monitor using AWS glue job run insights and AWS cloudtrail define workflows for ETL and integration activities AWS GLue Data Catalog

contains references to dat athat is used as source/target for ETL jobs to create data warehouse or data lake, must catalog this data data catalog → index to location, schema, runtime metrics of my data info stored as metadata tabkes, each table specify a single store AWS Glue databases

used to organize metadata tables in AS glue when we define a table in AWS glue data catalog, we add it to database database contains table sthat define data from many diff data stores partitioned table AWS Glue connection → data catelog object that stores login credentials, uri strings, and vpc info for particular data store

documentdb, opensearch, glue for spark, redshift, kafka etc etc Glue interactive sessions

quickly build test run data prep and analytics apps GUI for ETL stuff Data Processing Units (DPU)

monitoring section of AWS glue console uses results of previous job runs to determine appropriate DPU capacity use job metrics in AWS glue to estimate number of DPUs that can be used to scale out an AWS glue job Monitoring AWS Glue using Amazon Cloudwatch Metrics profile n monitor glue operations using glue job profiler collects n processes raw data into readable, near real-time metrics stored in cloudwatch → access historical info for analytics Amazon Macie Macie findings

repo of findings for macie account finding → detailed report of potential issue each finding → severity rating, info about affected resource + when and how macie found the issue More Kinesis Stuff ProvisionedThroughputExceededException → caused by capacity quota of data stream exceeding provisioned amount.

capacity quote defined by number of shards enhanced fan-out → scales up number of stream consumers by offering each stream consumer its own read throughput increase number of shards → capacity increases Resharding a stream → adjust number of shards

2 types of resharding operationr 1) shard split 2) shard merge shard split → divide single shard into 2 shards shard merge → 2 shard become 1 shard split → increases number of shards in stream → increase cost merge → decrease number of shards in stream → decrease $$ Troubleshooting Kinesis Data Streams Consumers

KDS records skipped → most common cause is unhandled exceptions thrown → to avoid infinite retries on recurring failure, KCL doesnt resend batch of records Record belonging to same shard processed by different record processors at same time → set appropriate failover time consumer app reading at slow rate → increase number of shards Appflow fully-manageed integration service that allows us to securely exchange data between SAAS apps and AWS services

Appflow Flows

a flow transfers data between a source and a destination data mapping determines how data from source is placed in dest Amazon Managed Workflows For Apache Airflow (MWAA) Apache Airflow → open source tool to programmatically author, schedule, monitor sequences of processes and tasks (workflows)

MWAA

managed orchestration service for apache airflow, can use to setup and operate data pipelines in cloud at scale use apache airflow+Python to create workflows without caring abt infra Some Container Storage Stuff app that transform data in containers (managed by AWS EKS). each containerized app will transform independent datasets and store data in data lake + data desont need to be shared to other containers. Which solution meets this reqs w lowest latency?

CORRECT → containers should use ephemeral volume provided by node's RAM WRONG → containers should establish connection to DAX w app code WRONG → containers should use PersistantVolume object provided by NFS storage WRONG → containers should establihs connection to amazon MemoryDB for Redis within app code Ephemeral Volume → storage local to each instance (lowest latency)

Some ML Stuff SageMaker Feature Store

feature store → storage and data maangement layer for ML. single source of truth to store/retrieve/remove/track/share/discover/control access to features online store → low latency, high avail store for feature group that enables real time lookup of recorsd offline store → stores historical data in S2 bucket. used when low latency reads are not needed feature group → logical group of features used to describe records → we can use sagemaker feature store to create, store, share features for ML model, but no native functinoality to ensure data is accurate, complete, trustworthy SageMaker ML Lineage Tracking creates n stores info about steps to ML workflow ability to establish model governance and audit standards ensure data being used to run ML decisions is accurate,complete, trustworthy SageMaker Data Wrangler → run EDA and prep data (but no native functionality to ensure data is accurate complete trustworthy)

SageMaker processing → managed servie we can use to run operations eg. data-processing workloads, data validation, model eval

More Glue Stuff AWS Glue DataBrew → data sanitiazation and normalization, can be used to remove outliers (some parameters to be specified) AWS Glue Schema Registry → data discovery features of AWS glue, integrate number of AWS servives to entrally manage and enforce schemas and supports streaming data sources. does no have an exclude_outliers parameter AWS Glue FindMatches → used to deduplicate a dataset AWS Glue Data Catalog → used to store metadata on data sources AWS Glue Spigot transformation → write ssample records to specified destination to help verify transformation performed by glue job AWS Glue Job w Pushdown Predicate → enables us to filter data in AWS glue data catalog by a partition AWS Glue SelectFields transform → creates new DynamicFrame from existing DynamicFrame, keeps only fields that we specify Athena Create Table As Select (CTAS) query creates new table in athena from results of SELECT statement stores data files created by CTAS statement in specified S3 loc create tables from query results in one step, without repeat Kinesis Firehose VS Kinesis Data Streams firehose can deiver streaming data directly to S2 KDS cannot directly — require intermediate stage

- quicksight has anomaly insights feature Amazon Aurora Fault Injection Test instance crash → ALTER SYSTEM CRASH Test replica failure → ALTER SYSTEM SIMULATE 50 PRECENT READ REPLICA FAILURE Test disk failure → ALTER SYSTEM SIMULATE 5 PERCENT DISK FAILURE Test disk congestion → ALTER SYSTEM SIMULATE 50 PERCENT DISK CONGESTION Lake Formation Stuff Lake Formation provides access according to user personas in conjunction with IAM permissions. Athena supports data anonymization using hashing Lake Formation Personas

IAM administrator → superuser, user who can create IAM users and roles. all permissions. CANNOT grant lake formation permissions if not also designated a data lake administrator Data Lake Administrator → can register S3 locations, access data catalog, create db, create and run worflows, grant LF permissions to others readonly admin → view only data engineeer → create db, create n run crawlers and workflows, grant lake formation permissions on data catalog tables that crawlers and workflows create. data analyst → can run queries against data lake using eg athena. can only RUN queries workflow → runs a workflow on behalf of user Cloudwatch Cloudtrail cloudtrail trail with cloudwatch logs enabled cloudwatch metric filter to select __ API entries in cloudtrail logs use metric filter to create cloudwatch alarm Some Terminology

Cloudwatch logs → monitor, store, access log files from EC2 instances, Cloudtrail, Route 53 and others Cloudwatch metric filter → filter some metrics Cloudtrail Insights → analyze normal patterns of API calling, generates insights events when volume/error rate are outside normal patterns STuff DynamoDB projection expressions → string that identified attributes that we want. To retrieve a single attribute, specify its name; for multiple attributes, names must be comma separated

Athena Federated Query

if have data sources other than S3, use Athena Federated Query to query data, or build pipelines then extract to S3 allows data to be queried in place using SQL Apache Flink

open-source, stream-processing, batch-processing framework distributed streaming data-flow engine executes arbitrary dataflow programs in parallel and pipeline Continuously Copy SQL Data To Redshift Use

AWS Database Migration Service (DMS) for database replication write transactions to S3 as unpartitioned parquet files create partitioned and sorted parquet files using AWS Glue Job run crawler to update Glue data catalog Load the target table using Amazon Redshift stored procedure Parquet files CSV problems

no defined schema lots of escaping eg. stuff with commas slow to parse cus row files cannot be filterred (no predicate pushdown) intermediate use requires redefining schema (lots of work) Parquet

columnar data format supported in spark, etc supports predicate pushdown, perf better automatically stores schema info binary file format Working with Pqrquet

spark.read.format('parquet').load(filename) spark.read.parquet(filename) df.write.parquet(filename) Partitioned vs unpartitioned parquet

by dividing data into smaller chunks, partitioning can improve performance of queries + reduce storage costs AWS CodeCommit source control service that hosts private Git repos. eliminates need for us to manage our own source control system, or worry about scaling infra like a manged git repo AWS CodePipeline continuous delivery service we can use to modeul, visualize, automate steps required to release software CICD stuff AWS CodeDeploy deployment service enables devs to automate deployment of apps to instances + update apps when required Creating pipeline to automate deployment of cloudformation template create pipline in AWS codepipeline using defualt location for artifact store add a source stage, and set provider as primary codecommit repo skip build stage add deploy stage with cloudformation as provide, and action as "create or update a stack" in output file name, enter "outputs" use parameter overrides field to enter documentDB required info edit deploy stage in pipeline to include deployment action that will create a change set for previously deployed stack Tracking db connections and exact queries executed enable audit logging using Redshift console, and configure logging on associated S3 bucket. Analyze the logs using Redshift Spectrum Redshift database audit logging

connection log — authentications attempts, connections, disconnections user log — changes to database use definitions user activity log — logs each query before it runs on db S3 preventing people from deleting stuff MFA delete — prevent accidental deletes

Object lock — prevent objects from being deleted or overwritten for fixed amount of time. Uses Write-Once-Read-Many (WORM) model

retention period → specified time period which object remains locked legal hold → same protection as retention period, but no expiration date object lock works only in buckets with S3 versioning enabled 2 retention modes

compliance mode → protected object version cannot be overwritten or deleted by any user including root user. retention mode cannot be changed, retention period cannot be shortened governance mode → users cant overwrite/delete object version or alter lock settings, unless user has special permission (less secure) Legal hold → prevent object deletion/overwrite until legal hold is removed

S3 expiring older versions Noncurrent version expiration

specify when noncurrent object versions expire after expire → permanently delete NewerNoncurrentVersions → how many newer noncurrent versions must exist before can delete NoncurrentDays → how many days an object is noncurrent before delete Noncurrent version transition specify when to move noncurrent version to S3 infrequent, glacier etc More Glue Stuff dataset in S3, Glue to perform transformations. How to prep data to be ingested from DynamicFrame to redshift cluster?

use Relationalize transform to change dat format can be used to convert data in a DynamicFrame into relational data format once relationalized, data can be written to red shift cluster READ MORE: GlueTransform

Real-time Stream Processing Approach Continuously ingest app log data, process it and write it to data lake + need real-time stream processing approach

Amazon managed streaming for Apache Kafka cluster (MSK). Create a Amazon managed service for Apache Flink app to read from MSK, process data and write output to data lake Transparent Data Encryption service available for oracle and SQL server databases for oracle, it is integrated with AWS CloudHSM, which is single-tenant hardware security module for AWS cloud AWS CloudHSM combines cloud w security of Hardware Security Modules (HSMs) HSM → computing device that processes cryptographic operations, provides secure storage for cryptographic keys AWS cloudHSM → control over HSMs on the cloud Pay by the hour Uses cases

AWS cloudHSM to manage private keys that protect highly confidential data → enterprise security standards encrypt n decrypt data → encryption in transit and at rest using private key to sign a document allows recipient to VERIFY that I sent that document. Can do this with CloudHSM private/public keys Cipher Message Authentication Codes (CMACs) and Hash Based Message Authentication Codes (HMACs) are used to authenticate and ensure integrity of messages sent over unsafe networks. with CloudHSM, can securely create and manage symmetric keys that support CMACs and HMACs Combine CloudHSM and KMS to store key material in SINGLE-TENANT environment → CloudHSM key stores To send data securely over HTTP, web servers use public/private key pairs and SSL/TLS public key certs to make HTTPS. CloudHSM can reduce computational burden → SSL/TLS offload Enable Transparent Data Encryption (TDE) → used to encrypt database files. encrypts data b4 storing on disk. Amazon DynamoDB Encryption Client enables you to protect your data in transit and at rest end to end protection for data from sources to destination Some Notes on AWS Step Functions serverless orchestration service where we can combine lambda w other AWS services to build apps through visual workflows Amazon States Language

JSON based structured language to define state machine (a collection of states) that can do work (Task states), determine which states to transition to next (Choice states) etc Types of states

Pass state → pass input to output without performing work (do nothing) Task state → represents a single unit of work performed by state machine. Uses activity OR lambda function to perform work Choice state → adds conditional logic to state machine Wait State → makes state machine wait for some time Success state → stops execution successfully, useful w choice states Fail state → stops exec, mark as failure, useful w choice states Parellel Sate → add separate branches of execution Map State → run set of workflow steps for each item in dataset Some Notes — Amazon Managed Streaming For Apache Kafka (MSK) fully managed service, enables us to build/run apps that use kafka to process streaming data. provides control-plane operations eg. create/update/delete clusters Some Terminology

producers give data to kafka broker consumers consume data from kafka broker multiple brokers in case one broker goes down messages auto replicated from one broker to another in case one die zookeeper used to manage bunch of brokers have multiple zookeepers, each coordinating between different brokers MSK Serverless possible to run Apache Kafka w/o needing to manage and scale cluster capacity — auto provisions and scale capacity for us More Redshift Notes fully managed, petabyte-scale data warehouse service in cloud users can analyze data using standard SQL and existing BI tools Columnar storage → faster query performance and efficient storage Node types → Dense compute VS Dense Storage

Updating Lambda functions easily eg Python script to do stuff, but if Python script change, engineer needs to update multiple lambda functions → need less manual way to do this

package custom Python script into Lambda layers. Apply Lambda layers to the lambda functions Lambda layers

a zip file archive that contains supplmentary code or data usually contain library dependencies, custom runtime or config Why use lambda layers

reduce size of deployment packages → instead of including all function dependencies in deployment packages, put them in layer separate core function logic from dependencies → update function dependencies independent of function code n vice versa share dependencies across multiple functions → after create layer, can apply it to any number of functions in account. Some ETL Stuff Step Functions → Not really for ETL Glue studio → ETL, but doesn't handle data pipeline MWAA → can use, but managed service, complex n possibly $$$ Glue workflows Amazon Redshift Data API suitable for real time queries access amazon redshift database using built-in redshift data api → can use lambda, sagemaker notebooks, cloud9 doesn't require persistent connection, provides secure HTTP endpoint and integration with AWS SDKs Athena permission controls need permission controls to seperate query processes and access to query history among users, teams and apps in the same AWS account

Create Athena workgroup for each use case. Apply tags to workgroup. Create IAM policy that uses tags to apply appropriate permission to WG

Athena Workgroup

recommend using workgroups to isolate queries for teams, apps or different workloads enforce cost constraints eg. per-query limit + per-workgroup limit track query-related metric for all workgroup AWS Glue Jobs Costs Glue Flex Execution

Flex → Glue jobs run on spare capacity reduce cost of pre-production, test, non-urgent data integration workloads by up to 34% good if dont need fast job start times (spare capacity not always avail) Glue Standard Execution

use mix of reserved and on-demand instances not as cost effective as flex execution Amazon Resource Name uniquely identify AWS resources, require ARN when need to specify a resource unambiguously acorss all AWS arn:partition:service:region:account-id:resource-id arn:partition:service:region:account-id:resource-type/resource-id arn:partition:service:region:account-id:resource-type:resource-id Compression in Athena snappy → quick compress/decompress rather than size of compress LZ4 → also focus on compression speed rather than size gzip → default write compression Amazon Managed Service for Apache Flink can use java/scala/python/sql to process and analyze streaming data can author/run code against streaming sources n static sources to perform time-series analytics 2 options for running Flink jobs

managed service for apache flink managed service for apache flink studio Redshift Concurrency Scaling supports 1000s of concurrent users and concurrent queries when turn on, auto add additional cluster capacity → turn on concurrency scaling for workload management (WLM) queue supports read/write/copy/insert/delete/update/create table statements supports materialized view refresh for MVs w no aggregation NOTE

WLM in redshift serverless no concurrency scaling WLM in redshift provisioned → have concurrency scaling Amazon Elastic Map Reduce Apache Pig, Oozie, Spark, Hbase, Flink process petabytes of data in a few seconds ETL → Amazon EMR 3 kinds of Amazon EMR

Amazon EMR running on Amazon EC2 Amazon EMR on EKS Amazon EMR serverless Amazon EMR → managed cluster platform that simplifies running big data frameworks eg. hadoop, spark → process vast amounts of data

Nodes in Amazon EMR

primary node → node that manages cluster, coordinates distribution of data n tasks among other nodes for processing. tracks status of tasks, monitor health of cluster. core node → node w software components that run tasks and store data in hadoop distributed file system on your cluster. task node → node with software components that only run tasks, does not store data in HDFS (this is optional) Glue Workflows use workflows to create n visualize complex ETL activities involving multiple crawlers, jobs, triggers. visual representation of workflow as a graph can create from AWS Glue blueprint, OR manually build Triggers in Glue Workflow

schedule → workflows starts according to schedule we define on demand → we manually start it EventBridge event → workflow starts from event More ETL Stuff multiple ETL workflows that ingest data from db to S3 data lake use AWS Glue and Amazon EMR to process data company wants to improve existing archi to provide automated orchestration which requires minimal manual effort → AWS Glue Workflows

Redshift Data Sharing securely share access to live data across redshift clusters, workgroups, AWS accounts, AWS regions no need to manually move or copy data We can share data at different levels in Amazon redshift

database, schema, table, view levels etc Redshift Spectrum we can query and retrieve structured and semistructured data from S3 use massive parallelism to run very fast against large datasets AWS Graviton graviton processors are custom-built by AWS to deliver best price performance for cloud workloads offer up to 40% better price performance S3 partitions large orgs have lots of data → usually stored in S3 Partitioning → lay out data optimally for distributed analytics engines

restrict amount of data scanned → improve performance + reduce cost eg if data is partitioned by day, we get 1 single file every day s3://mybucket/logs/year=x/month=6/day=1/test.json s3://mybucket/logs/year=x/month=6/day=2/test.json s3://mybucket/logs/year=x/month=6/day=3/test.json s3://mybucket/logs/year=x/month=6/day=4/test.json Select * rfom table where year=x and month=6 and day=1 → reads only data inside the partition folder year=x/month=6/day=1 → only scans this

systems like Athena, Redshift Spectrum and Glue can use these partitions to filter data by value → improve performance AWS Glue Partition Index over time, many many partitions are added to a table when create partition index, we specify list of partition keys that already exist on a given table partition index can be created on permutation of partition keys defined on table eg. (country, category, year) data catalog will concatenate partition values in order Athena Partitioning and bucketing mainly to reduce amount of data athena needs to scan when querying Partitioning

organizing data into directories (or prefixes) on S3 based on particular properties of data such properties are known as partition keys eg date, year, month good candidates for partition keys → properties that are frequently used in queries and have low cardinality Bucketing

way of organizing records of dataset into categories called buckets records with same value for a property go into same bucket records are distributed as evenly as possible among buckets Conclusion I hope this was helpful in some way