CloudNinja.pro

ASSOCIATE · DEA-C01

# AWS Certified Data Engineer – Associate

## DEA-C01 Cheat Sheet

CloudNinja · cloudninja.pro · Free — no signup required

Every service in scope, every batch and streaming pattern, every storage-format, governance and orchestration decision the DEA-C01 exam actually tests — organised around the four exam domains. Assumes ~ 2 years of data engineering experience.

### EXAM SNAPSHOT

- **65 questions** — 50 scored + 15 unscored
- **130 minutes** total time
- **720 / 1000** to pass
- **$150 USD** exam fee

- **Ingestion & Transformation** — 34%
- **Data Store Management** — 26%
- **Operations & Support** — 22%
- **Security & Governance** — 18%

FREE DEA-C01 practice exam →

# 1 · Data Engineering Fundamentals

- **Structured vs semi-structured vs unstructured** — tables (row / column) vs JSON / XML / Parquet vs blobs (images, audio, video).
- **OLTP vs OLAP** — operational row-store (RDS, Aurora, DynamoDB) vs analytical column-store (Redshift, Athena on Parquet, Spark on Iceberg).
- **Batch vs streaming** — hourly/daily/weekly jobs vs sub-second continuous ingest. Many pipelines do both (lambda / kappa architectures).
- **Data lake vs warehouse vs lakehouse** — lake = raw, schema-on-read (S3); warehouse = modeled, schema-on-write (Redshift); lakehouse = ACID table format (Iceberg, Hudi, Delta) over S3 + engines (Athena, EMR, Redshift Spectrum) on top.
- **ETL vs ELT** — Extract → Transform → Load (Glue, Spark jobs) vs Extract → Load → Transform (dbt-style on Redshift / Athena). ELT defers modeling to the warehouse.
- **Medallion architecture** — Bronze (raw) → Silver (cleansed / joined) → Gold (business-ready). Encode as S3 prefixes + table versions.
- **Partitioning vs bucketing** — partitioning = separate files/paths per key value (skip-scan); bucketing = hash-distributed files per key (join locality).

### Columnar Formats

| Format | When |
|---|---|
| **Parquet** | Default analytical choice. Columnar, splittable, compressed (Snappy/ZSTD), schema embedded. |
| **ORC** | Hive ecosystems; slightly better compression on some workloads. |
| **Avro** | Row-oriented with rich schema evolution — stream / event data (Kinesis / MSK + Schema Registry). |
| **JSON / CSV** | Ingest + interchange only. Don't query at scale. |

# 2 · Ingestion Services

| Service | Role |
|---|---|
| **Amazon Kinesis Data Streams** | Durable sharded stream; 24 h – 365 d retention; at-least-once. Producers: KPL, Kinesis Agent, SDK. Consumers: KCL, Lambda, Flink; *Enhanced Fan-Out* = 2 MB/s per consumer. |
| **Amazon Kinesis Data Firehose** | Managed delivery to S3, Redshift, OpenSearch, Splunk, HTTP. Buffering by size / interval; Lambda transformation; Parquet / ORC conversion; dynamic partitioning. |
| **Amazon Managed Streaming for Apache Kafka (MSK)** | Managed Kafka — brokers, Zookeeper / KRaft, IAM + SASL/SCRAM + TLS auth; MSK Serverless; MSK Connect for source/sink connectors. |
| **Amazon Managed Service for Apache Flink** | Stream processing — SQL, Table API, DataStream; stateful aggregations, windows (tumbling, sliding, session), CEP. Replaces Kinesis Data Analytics. |
| **AWS DataSync** | Bulk / incremental file transfer (NFS, SMB, HDFS, S3, EFS, FSx). TLS; bandwidth throttling; validation. |
| **AWS Database Migration Service (DMS)** | Homogeneous + heterogeneous DB migration; CDC streaming to S3 / Kinesis / Redshift / Kafka. Serverless DMS + SCT for schema conversion. |
| **AWS Transfer Family** | Managed SFTP / FTPS / FTP / AS2 / WebApp; lands partner data in S3 / EFS. |
| **AWS Application Migration Service / Application Discovery Service** | Server migration with replication + agent; used when whole apps (+ their data) move to AWS. |

| Service | Role |
|---|---|
| **AWS Snow Family** | Petabyte-scale physical transfer when network is impractical (legacy — new customer signup ending; for exam: PB-scale offline ingest). |
| **Amazon AppFlow** | Managed SaaS → AWS connectors (Salesforce, Zendesk, Slack, SAP, Marketo, ServiceNow) into S3 / Redshift / Snowflake. |

### Kinesis Data Streams — Key Numbers

- Shard: 1 MB/s or 1,000 rec/s in, 2 MB/s out (shared). Per consumer in EFO: 2 MB/s dedicated.
- Record max 1 MB; PartitionKey determines shard.
- On-demand mode auto-scales shards; provisioned requires manual *resharding* (split / merge) when hot-shard limits hit.
- Checkpointing via KCL + DynamoDB lease table; Lambda uses its own checkpointer (*ParallelizationFactor* up to 10 per shard).

## 3 · Transformation & Orchestration

| Service | Use |
|---|---|
| **AWS Glue** | Serverless Spark + Python shell ETL. *Data Catalog* (metastore), *Crawlers*, *Jobs* (Spark, Streaming, Ray, Python Shell), *Workflows*, *Triggers*, *Bookmarks* (incremental tracking), *DataBrew* (low-code prep), *Data Quality* (rules via DQDL), *Schema Registry*, *Sensitive Data Detection*. |
| **AWS Glue DataBrew** | Low/no-code visual data prep; 250+ transformations; profile + lineage; output to S3 / Glue Catalog. |
| **Amazon EMR** | Managed Spark / Hive / Presto / Trino / HBase / Flink. On-cluster EC2 (task / core / master) or *EMR Serverless* (on-demand Spark/Hive without cluster mgmt). *EMR on EKS* runs on your cluster. Use Spot for task nodes. |
| **AWS Lambda** | Event-driven transformations (S3 PUT, DynamoDB Streams, Kinesis). Limits: 15 min, 10 GB mem, 10 GB `/tmp`. SDK + SAM for quick pipelines. |
| **AWS Batch** | Managed batch jobs on Fargate / EC2 / EKS; job queues, compute envs, array jobs for fan-out. |
| **Amazon ECS · Amazon EKS · Amazon ECR** | Containers for custom ETL workers and ML pipelines. ECS + Fargate (serverless) or ECS on EC2 for GPU jobs; EKS + Karpenter when you already run Kubernetes; ECR for private image registry with Enhanced scanning via Inspector. Glue, EMR on EKS and SageMaker Processing can all consume custom images from ECR. |
| **AWS Step Functions** | Durable orchestration: Standard (< 1 y) and Express (< 5 min). Native integrations with Glue, EMR, Redshift Data API, Lambda, Athena, SageMaker. Map (including distributed Map up to 10k executions). |
| **Amazon MWAA** | Managed Airflow for complex DAG orchestration — first answer when teams already use Airflow or when Python-first DAGs are required. |
| **Amazon EventBridge** | Event bus (default + custom + partner) + Pipes (source → filter → enrich → target) + Scheduler. Great for cron + fan-out pipeline triggers. |
| **Amazon SNS / SQS** | Pub/sub + durable queue. SQS FIFO for ordering / dedupe; DLQ for failed messages; extended client for > 256 KB payloads in S3. |

**Exam cue:** "Visual DAG orchestration across multiple services with retries + state" → Step Functions (or MWAA if Airflow-native). "Schedule cron across many accounts" → EventBridge Scheduler. "Simple S3-PUT triggered conversion" → Lambda direct.

# 4 · Storage for Analytics

## Amazon S3 — Analytical Data Lake

- **Storage classes**: Standard, Intelligent-Tiering (auto), Standard-IA, One Zone-IA, Glacier Instant / Flexible / Deep Archive. Lifecycle policies transition / expire.
- **Event Notifications** → SQS / SNS / Lambda / EventBridge for reactive pipelines.
- **Versioning + Object Lock** (WORM) — ransomware + compliance.
- **Replication** — SRR / CRR with Replication Time Control (15-min SLA).
- **Access Points** + Multi-Region Access Points for per-workload + global routing.
- **S3 Select** CLOSED TO NEW — closed to new customers Jul 2024. Use Athena instead.
- **Amazon S3 Tables** — managed Apache Iceberg tables in S3 with per-table bucket optimisation (compaction, snapshot management); integrates with Athena, EMR, Redshift Spectrum. First answer for "data lake with ACID transactions + schema evolution + time-travel".
- **Amazon S3 Glacier** — cold / archive. Vault Lock for compliance retention.

## Other Storage

- **Amazon EBS** — block, single-AZ; *gp3* default, *io2 Block Express* for high IOPS.
- **Amazon EFS** — shared NFS multi-AZ; used as scratch / home dir for SageMaker / EMR Studios.
- **AWS Backup** — central backup for EBS / EFS / RDS / Aurora / DynamoDB / FSx / EC2 / S3 / Storage Gateway; org-wide plans via delegated admin.

# 5 · Analytical Engines & Warehouses

## Amazon Athena

- Serverless SQL on S3. Presto / Trino SQL; Athena engine v3 default.
- **Cost** = $/TB scanned. Reduce by Parquet + partitioning + bucketing + column pruning + compression. Result reuse + reuse-if-available.
- **Workgroups** — separate cost + enforce encryption + per-query data scanned limit.
- **Federated Query** via Lambda connectors to non-S3 sources (DynamoDB, RDS, Redshift, CloudWatch Logs).
- **Athena for Apache Spark** — serverless PySpark notebooks.
- **CTAS / UNLOAD** — create new partitioned Parquet data sets from queries.

## Amazon Redshift

- **Nodes** — RA3 (managed storage, decoupled compute/storage), DC2 (fixed local SSD). Concurrency Scaling adds transient clusters for burst.
- **Redshift Serverless** — pay per RPU-second; auto-pause + resume.
- **Table design** — *DISTKEY* (even / key / all), *SORTKEY* (compound / interleaved), *ENCODE* (AUTO, ZSTD, AZ64). VACUUM + ANALYZE maintain performance; auto-vacuum on by default.
- **Spectrum** — query S3 directly (incl. Iceberg); extend warehouse without loading.
- **Zero-ETL** — Aurora / RDS MySQL / PostgreSQL → Redshift near-real-time; DynamoDB → Redshift; OpenSearch → Redshift.
- **Data sharing** — share live tables across accounts / Regions without copying.
- **Redshift ML** — invoke SageMaker / Bedrock via SQL.
- **Materialized views + Auto Materialized Views** — speed up repeated queries.

## Amazon EMR Engines

- **Spark** — default. EMR Serverless for on-demand, EMR on EC2 for Spot / custom AMIs, EMR on EKS for shared infra.
- **Hive** — legacy metadata + batch SQL (Tez).
- **Presto / Trino** — interactive SQL across S3 + HDFS + JDBC.

- **HBase** — NoSQL on top of S3 (EMRFS).
- **Flink** — streaming (or use Managed Flink).
- **Table formats** — Iceberg (recommended), Hudi, Delta Lake with supported engines.

### Amazon OpenSearch Service

- Search + log analytics. UltraWarm + Cold tiers for cost; anomaly detection + alerting; SAML SSO; domains + Serverless.
- **Ingestion** — OpenSearch Ingestion (Data Prepper), Firehose, Logstash, custom.
- Vector search via k-NN plugin; Neural plugin integrates with Bedrock.

## 6 · Data Catalog, Lakehouse & Governance

### AWS Glue Data Catalog

- Central metastore used by Athena, EMR, Redshift Spectrum, Lake Formation, SageMaker.
- **Crawlers** — infer schema + create / update tables + partitions. Schedule daily or trigger on S3 event.
- **Partition projection** (Athena) avoids catalog partition explosion for time-based datasets.

### AWS Lake Formation

- Fine-grained access control (FGAC) on Data Catalog — **row-level**, **column-level**, **cell-level**, tag-based (LF-Tags).
- Centralised permissions replace S3 bucket policies for analytical access (hybrid mode for gradual migration).
- Cross-account + cross-Region data sharing; federated identities via IAM Identity Center.
- Governed tables (legacy) — prefer Iceberg / S3 Tables for new.

### AWS Data Exchange

- Subscribe to third-party data products as managed S3 datasets, Redshift data shares, API, or Lake Formation grants.
- Useful for enriching pipelines with market / demographic / satellite data.

## 7 · Databases — OLTP & Purpose-Built

| Service | Use |
|---|---|
| **Amazon RDS** | Managed MySQL / PostgreSQL / MariaDB / Oracle / SQL Server / Db2. Multi-AZ, read replicas, automated backups, Performance Insights, Blue/Green deployments. |
| **Amazon Aurora** | MySQL/PostgreSQL-compatible; storage auto-grows to 128 TiB; 15 read replicas; Global Database (sub-second cross-Region); Aurora Serverless v2 for variable workload; *Zero-ETL* to Redshift. |
| **Amazon DynamoDB** | Key-value + document, single-digit-ms. On-demand or provisioned + auto-scaling. GSI / LSI, Streams → Lambda, TTL, Transactions, PITR + on-demand backup, Global Tables for multi-Region active-active. |
| **Amazon DocumentDB** | MongoDB-compatible managed document store. |
| **Amazon Keyspaces** | Apache Cassandra compatible; serverless, integrated with IAM. |
| **Amazon MemoryDB for Redis** | Redis-compatible durable primary database (Multi-AZ transaction log). |
| **Amazon Neptune** | Graph (Gremlin / SPARQL / openCypher); Neptune Analytics for vector + graph. |

## 8 · Visualisation, BI & ML-in-Pipeline

- **Amazon QuickSuite** — AWS's BI + AI workspace (2025 rebrand unifying QuickSight, Q, and related tools). SPICE in-memory engine for fast dashboards; Q for natural-language BI; ML-powered narratives + anomaly detection; row-level +

column-level security; embedded analytics via SDK.

- **Amazon Managed Grafana** — dashboards across CloudWatch, OpenSearch, Prometheus, X-Ray, Timestream, Athena, Redshift.
- **Amazon SageMaker AI** — build / train / deploy ML models. From a data-engineering lens: Feature Store, Data Wrangler, Processing jobs + Pipelines to integrate ML into ETL flows.
- **Amazon Bedrock** — foundation models for data enrichment (classification, summarisation, entity extraction); Titan Embeddings for vector columns; Guardrails for PII redaction.
- **Amazon Kendra** — semantic search over S3 / SharePoint / Confluence / RDS; pair with Bedrock for RAG.
- **Amazon Q** — enterprise AI assistant used by Q in QuickSuite / Q Business / Q Developer.

# 9 · Streaming & Real-Time Patterns

### Typical architectures

- **Lambda architecture** — batch (Glue / EMR on S3 history) + speed (Flink / Lambda on Kinesis). Serve merged view via Athena / Redshift.
- **Kappa architecture** — single streaming path (Kinesis / MSK → Managed Flink → Redshift / S3 Iceberg).
- **Change Data Capture** — RDS / Aurora via DMS CDC → Kinesis / MSK → Iceberg; or Aurora Zero-ETL to Redshift.
- **Event-driven lake** — S3 PUT → EventBridge → Step Functions → Glue job → Athena-queryable partition.
- **Backfill** pattern — ingest full snapshot once, then CDC; merge via Iceberg MERGE.

### Exactly-once + ordering notes

- Kinesis Data Streams = at-least-once; dedupe on downstream via idempotent writes (Iceberg MERGE, DynamoDB conditional PutItem).
- Kafka + idempotent producer + transactions = exactly-once within a single application.
- SQS FIFO = exactly-once processing within message group; 3000 msgs/s with batching.

# 10 · Security, Identity & Compliance

| Service | Role |
|---|---|
| **AWS IAM** | Roles for Glue / EMR / Lambda / SageMaker execution. Least privilege; `aws:PrincipalOrgID`, `aws:SourceArn`, `aws:SecureTransport`. |
| **AWS KMS** | Customer-managed keys for S3, EBS, Redshift, RDS, Aurora, DynamoDB, Firehose, CloudWatch Logs. `kms:ViaService` to scope. |
| **AWS Secrets Manager** | Rotate DB creds, third-party API keys; accessed by Glue / Lambda / EMR at runtime. |
| **Amazon Macie** | Discover PII / PHI / regulated data in S3; alerts drive bucket policy + tagging remediation. |
| **AWS WAF / AWS Shield** | Protect API Gateway + CloudFront + ALB fronting data-intake APIs from L7 attacks and DDoS. |
| **AWS CloudTrail** | Audit every data API call (Glue, Athena, Redshift Data API). Org trail to Log Archive. |
| **AWS Config** | Resource compliance (*s3-bucket-server-side-encryption-enabled*, *redshift-cluster-configuration-check*) + auto-remediation via SSM Automation. |

### Encryption patterns

- At rest — KMS CMK everywhere; S3 bucket policy *Deny* unless SSE-KMS with specific key.
- In transit — TLS 1.2+; RDS / Aurora via `rds.force_ssl`; Redshift with `require_ssl`.
- Client-side — AWS Encryption SDK / S3 Encryption Client for end-to-end envelope encryption.
- Column-level / row-level filtering via Lake Formation; dynamic data masking in Redshift.

# 11 · Networking for Data Pipelines

- **Amazon VPC** — private subnets for data stores (RDS, Redshift, OpenSearch, EMR). Security groups + NACLs.
- **VPC Gateway endpoint (S3 / DynamoDB)** — free; avoids NAT egress on massive pulls.
- **VPC Interface endpoint (PrivateLink)** for Glue, Athena, Kinesis, MSK, KMS, Secrets Manager, SageMaker — keeps data-plane traffic private.
- **Amazon Route 53** — DNS for private endpoints; weighted/latency routing for multi-Region data services.
- **Amazon CloudFront** — CDN front for read-heavy static data / dashboards / S3 public assets.
- **Amazon API Gateway** — fronts data-intake / data-access APIs (REST/HTTP); WAF + usage plans; IAM / Cognito / Lambda authorizers.

# 12 · Observability, Reliability & Cost

| Service | Use |
|---|---|
| **Amazon CloudWatch** | Metrics: Glue *glue.driver.aggregate.bytesRead*, Kinesis *IteratorAgeMilliseconds*, Redshift *WLMQueueLength*, RDS *CPUUtilization*. Alarms (composite, anomaly, metric math). Dashboards + Synthetics. |
| **Amazon CloudWatch Logs** | Job logs (Glue, Lambda, EMR, Redshift audit logs, Kinesis errors). Logs Insights + subscription filters → Firehose → OpenSearch. |
| **Amazon Managed Grafana** | Unified dashboards across CloudWatch + Prometheus + Athena + Redshift + X-Ray. |
| **AWS Systems Manager** | Parameter Store for config (DB endpoints, job parameters), Patch Manager for EC2 data nodes, Change Manager for prod change windows, Incident Manager for on-call. |
| **AWS Well-Architected Tool** | Perform reviews with Data Analytics Lens / Machine Learning Lens. |
| **AWS Budgets / Cost Explorer** | Budgets + actions (stop dev EMR on overage); Cost Explorer for spending by service/usage-type. |

### Cost patterns that score on the exam

- **Athena** — Parquet + partitioning + compression; per-query data scanned limit in workgroup.
- **Redshift** — RA3 + Spectrum; Concurrency Scaling; Reserved Nodes or Redshift Serverless; compression + sort keys.
- **EMR** — EMR Serverless or Spot task nodes; auto-termination; Graviton instances.
- **S3** — Intelligent-Tiering; Lifecycle to IA / Glacier; request consolidation (multipart + CRT).
- **Kinesis** — on-demand mode avoids over-provisioning; reduce retention if not needed.
- **Glue** — DPU right-sizing; Auto Scaling; Flex execution (cheaper, slower) for non-urgent jobs.

# 13 · DevOps for Data Pipelines

- **AWS CloudFormation / AWS CDK** — IaC for Glue jobs + crawlers, Lake Formation grants, Redshift clusters, Kinesis streams, EMR clusters, Step Functions state machines.
- **AWS SAM** — serverless application model for Lambda + API GW + EventBridge + Step Functions.
- **AWS CLI** — scripting + CI/CD; use `aws s3 sync`, `aws glue start-job-run`, `aws athena start-query-execution`.
- **AWS CodeBuild / CodeDeploy / CodePipeline** — CI/CD for pipeline code + IaC. Promotion dev → staging → prod with data-quality gates (Glue Data Quality) as part of pipeline.
- **Amazon Q Developer** — AI coding assistant (IDE + CLI + console) for writing Spark / SQL / Glue code.

# 14 · Reference Architectures

### A · Serverless lakehouse

- Landing: Kinesis Firehose → S3 (Parquet, date-partitioned).
- Bronze → Silver: Glue Job (PySpark) reads raw, cleanses, writes Iceberg table (S3 Tables).
- Silver → Gold: Athena CTAS + Glue Workflows; Lake Formation permissions.
- Serve: QuickSuite dashboards on Athena; Redshift Spectrum for warehouse users.
- Orchestration: Step Functions triggered by EventBridge on S3 PUT.
- Observability: CloudWatch + Glue Data Quality + Budgets alarms.

### B · Streaming ETL with CDC

- Source: Aurora PostgreSQL via Zero-ETL to Redshift OR DMS CDC → Kinesis.
- Stream processing: Managed Flink applies enrichment + windowed aggregates → Firehose → S3 Iceberg.
- Dead-letter handling: SQS DLQ + CloudWatch alarm; Lambda redrive.
- Backfill: one-shot Glue job populates history; Iceberg MERGE reconciles.

### C · Enterprise data mesh

- Domain accounts own S3 + Glue Catalog + Lake Formation permissions.
- Central governance account runs Glue Data Quality + Macie + Config org-wide.
- Data product publishing via Lake Formation cross-account grants or Redshift data share.
- Discovery via SageMaker Unified Studio / QuickSuite / Q Business.

### D · Event-driven batch pipeline

- S3 PUT → EventBridge rule → Step Functions → Glue Job (Parquet conversion + partitioning) → Athena.
- Glue Data Quality evaluated pre-load; on fail, alert via SNS + Chatbot to Slack.

### E · Hybrid ingest for large legacy system

- DataSync / Transfer Family land files in S3; DMS replicates DB tables with CDC; Snow Family for the initial PB-scale seed; Application Migration Service for app replatform.
- Glue Crawler + Lake Formation grants expose new tables to analysts day-one.

# 15 · Scenario → Answer Patterns

| Scenario | Answer |
|---|---|
| Cheapest way to query JSON logs in S3 | Convert to Parquet, partition by date, query with Athena. |
| Row / column-level security on data lake | AWS Lake Formation permissions + LF-Tags. |
| Near-real-time ingest from Aurora to Redshift | Aurora Zero-ETL to Redshift. |
| Manage schema evolution for streaming data | Glue Schema Registry (Avro / JSON Schema / Protobuf). |
| ACID transactions on the data lake | Apache Iceberg via Amazon S3 Tables + Athena / EMR / Redshift Spectrum. |
| Classify PII in S3 before landing in warehouse | Amazon Macie; alerts + Glue Sensitive Data Detection as pre-load check. |
| Keep long-running Airflow DAGs | Amazon MWAA (managed Airflow). |
| Replace on-prem Hadoop | EMR Serverless or EMR on EC2 + S3 + Glue Catalog. |
| Load from SaaS app (Salesforce, Zendesk) into data lake | Amazon AppFlow (or DMS if DB source). |

| Scenario | Answer |
|---|---|
| Deduplicate streaming records | Idempotent writes via Iceberg MERGE or DynamoDB conditional PutItem; Kafka idempotent producer for in-pipeline. |
| Grant another account access to Redshift data without copying | Redshift data sharing. |
| Query DynamoDB + RDS + CloudWatch Logs together | Athena Federated Query with Lambda connectors. |
| Secure credentials rotating every 30 days | AWS Secrets Manager with rotation Lambda. |
| Detect late-arriving or malformed records | AWS Glue Data Quality (DQDL rules) + CloudWatch alarms + SNS. |
| Isolate analytics account from producer accounts | Account per domain, centralised Glue Catalog in analytics account + Lake Formation cross-account grants. |
| Replace Kinesis Data Analytics | Amazon Managed Service for Apache Flink. |
| Run batch containers with job queues + retries | AWS Batch (Fargate or EC2). |
| Reduce Athena bill for a single team | Dedicated workgroup with per-query data-scanned limit + enforced result location + Parquet conversion. |
| Warehouse query across 2 Regions without copying | Redshift data sharing across Regions; or Athena + Glue Catalog multi-Region with S3 CRR. |
| Visualise + natural-language query on top of warehouse | Amazon QuickSuite (with Q in QuickSuite). |
| Ensure pipeline can reprocess data if code changes | Glue Job Bookmarks disabled / reset; partitioned idempotent writes; Iceberg time-travel for historical restores. |
| Full lineage + data profile | AWS Glue DataBrew profile jobs + Glue Data Quality + Lake Formation audit + QuickSuite lineage. |

## 16 · Common Pitfalls

- **Firehose is not Kinesis Data Streams** — no consumer code, no replay, no custom consumers. Pick Streams when you need multiple consumers, replay, or > 60 s latency budget.
- **Glue Crawlers on large time-partitioned prefixes** explode the catalog — use partition projection on Athena instead.
- **Athena reads whole files** unless you use Parquet + column projection; CSV + gzip is slow + expensive.
- **Redshift WLM** — automatic WLM with query monitoring rules; Short Query Acceleration for mixed workloads.
- **DMS full-load + CDC** requires enough storage + network; plan replication instance size; task parallelism.
- **EMR Spot instances** — use only on task nodes; master / core on On-Demand for resilience.
- **Kinesis shard limits** — 1 MB/s in, 2 MB/s shared out; On-Demand is simpler.
- **Lake Formation vs IAM** — LF grants don't override explicit IAM deny; hybrid mode requires careful staging.
- **S3 Tables vs classic Iceberg on S3** — S3 Tables handles compaction + snapshot mgmt for you; DIY Iceberg requires Glue workflows for maintenance.
- **Zero-ETL latency** — near-real-time, not instant; typically single-digit seconds but varies by load.
- **Data sovereignty** — pin Region via SCP; ensure Glue / Athena / Firehose destinations stay in-region.
- **Athena cost explosion** — scans entire dataset without partitioning or without `LIMIT` pushdown for SELECT *; always filter on partition keys.

**DEA-C01 default answers:** "Cheap serverless SQL on S3" → Athena on Parquet + partitions. "Managed Spark" → EMR Serverless (or Glue for integrated catalog + bookmarks). "ACID data lake" → S3 Tables + Iceberg. "Fine-grained security" → Lake Formation. "Ship DB to warehouse" → Zero-ETL or DMS + Redshift. "Streaming transformation" → Managed Flink. "Schedule + orchestrate" → Step Functions or MWAA. "Surface data to analysts" → QuickSuite.

# 17 · AWS Glue Deep Dive

### Job types

| Job type | When |
|---|---|
| **Spark (batch)** | Classic ETL. DPU = 4 vCPU / 16 GB. Scala or Python (PySpark). DynamicFrames + DataFrame APIs. |
| **Spark Streaming** | Continuous Spark on Kinesis / MSK / Kafka. Checkpointing + watermarks. |
| **Python shell** | Lightweight single-node Python; cheap (0.0625 or 1 DPU). For small transforms, API calls, lightweight ML. |
| **Ray** | Distributed Python for ML workloads (not yet Spark-equivalent coverage). |

### Bookmarks, Workflows & Triggers

- **Bookmarks** — Glue tracks processed files / partitions; reset to reprocess.
- **Workflows** — DAG of jobs + crawlers + triggers in one UI.
- **Triggers** — on-demand, schedule, event (EventBridge), job-success/fail conditional.
- **Job metrics** — CloudWatch + Spark UI archived to S3.

### Glue Data Quality (DQDL)

- Declarative rules (`ColumnCount > 5`, `IsComplete "id"`, `IsUnique "id"`, `ColumnValues "status" in [ "A", "B" ]`, `ColumnLength "zip" between 5 and 10`, custom SQL).
- Evaluate in-flight (blocking) or async; publish results to EventBridge for alerting.
- Recommended rules auto-generated from data profile.

### Glue Studio & Interactive Sessions

- Glue Studio — visual ETL builder generating PySpark.
- Interactive Sessions — notebooks (Jupyter / VS Code) backed by Glue Spark without managing cluster.

# 18 · Redshift Performance Playbook

### Physical design

- **Distribution style** — *KEY* on frequent join column, *ALL* for small dimension tables (< a few million rows), *EVEN* as default, *AUTO* lets Redshift decide.
- **Sort keys** — *compound* for one-sided range filters; *interleaved* when filters rotate columns; usually pick compound on date.
- **Compression (encoding)** — AUTO usually correct; ZSTD strong default; AZ64 for numeric; RAW only for tiny tables.
- **VACUUM / ANALYZE** — Auto on by default; monitor *SVV_VACUUM_PROGRESS*.

### Ingest patterns

- **COPY** from S3 (Parquet, ORC, JSON, CSV) in parallel; use MAXERROR to skip bad rows; manifest for atomic file sets.
- **Auto-copy** from S3 — Redshift watches a folder.
- **Zero-ETL** sources (Aurora, RDS MySQL/PostgreSQL, DynamoDB, OpenSearch) — near-real-time, serverless, schema-aware.
- **Streaming Ingestion** — Kinesis Data Streams or MSK directly to Redshift materialized view.
- **Spectrum** — external table over S3; great for cold data or Iceberg tables.

### Concurrency + sharing

- **Automatic WLM** — default; query monitoring rules control runaway queries.
- **Concurrency Scaling** — spins transient clusters for read queues; burst without re-provisioning.

- **Data sharing** — share live tables cross-cluster / cross-account / cross-Region without copy.
- **Datashare with Lake Formation** — FGAC on shared data.
- **Redshift Serverless** — pay per RPU-second; auto-pause; Query Editor v2 for notebooks.

# 19 · Lakehouse Table Formats

### Apache Iceberg (preferred)

- ACID on S3 via manifest files; schema evolution (add / drop / rename); partition evolution; time travel (`FOR TIMESTAMP AS OF`).
- Supported engines: Athena (v3), EMR Spark / Trino, Redshift Spectrum (read + write), SageMaker.
- **Amazon S3 Tables** — fully-managed Iceberg: optimises compaction, snapshot expiration, unreferenced file cleanup automatically per table bucket.
- MERGE for upserts from CDC; row-level deletes via position or equality delete files.

### Apache Hudi / Delta Lake

- EMR supports both; Glue supports Hudi + Delta as managed connectors.
- Pick Hudi for record-level upserts with Copy-on-Write / Merge-on-Read; Delta when the rest of org is on Databricks.

# 20 · Data Pipeline Runbooks

### Glue job failure

1. Check CloudWatch Logs + Spark UI archive for stack trace.
2. If OOM: increase DPU or switch to G.2X / G.4X workers; repartition data.
3. If skew: salt key or broadcast small side.
4. If throttle on downstream (S3 / DDB): use exponential back-off, throttling-aware writer.
5. Reset bookmark if rerunning from scratch.

### Kinesis consumer lag

1. CloudWatch *IteratorAgeMilliseconds* increasing.
2. Enable Enhanced Fan-Out for dedicated 2 MB/s per consumer.
3. Increase Lambda *ParallelizationFactor* (up to 10) per shard.
4. Add shards (reshard) or switch to On-Demand mode.

### Redshift slow query

1. *STL_QUERY* / *SVL_QUERY_SUMMARY* + query plan.
2. Missing sort key or dist key? Rebuild with correct keys.
3. Stale stats? ANALYZE; check Auto Vacuum.
4. Concurrency issue? Monitor *WLMQueueLength*; Concurrency Scaling.
5. Consider materialized view or CTAS into aggregated table.

### Athena cost spike

1. Workgroup per-query scan limit.
2. Convert CSV → Parquet + partition on query-filter columns.
3. Compression + column pruning; avoid `SELECT *`.
4. Result reuse for repeated queries.
5. Investigate via *cloudtrail* + Athena query history with Cost Allocation tags.

## 21 · Data Quality, Lineage & Compliance

- **Profile** — Glue DataBrew profile job; Glue crawler schema inference; summary statistics.
- **Validate** — Glue Data Quality rules; pre-ingest gate in Step Functions; block pipeline on failure.
- **Classify** — Macie for PII + Glue Sensitive Data Detection; tag S3 objects / Catalog columns.
- **Lineage** — QuickSuite lineage views + DataZone (legacy) / SageMaker Unified Studio in newer setups.
- **Retention** — S3 Lifecycle to Glacier; Object Lock Compliance for regulated retention.
- **Audit** — CloudTrail Data Events on S3 + DDB; Athena queries over CloudTrail Lake; AWS Config rules (*s3-bucket-default-lock-enabled*, *redshift-cluster-kms-enabled*).

## 22 · Architectural Decision Cheatsheet

| Need | Pick |
|---|---|
| Batch ETL with integrated catalog + minimal ops | AWS Glue + Glue Data Catalog |
| Spark at scale with custom images | EMR Serverless or EMR on EC2 |
| Kafka API compatibility | Amazon MSK (or MSK Serverless) |
| Stream processing with windows + CEP | Managed Service for Apache Flink |
| SQL over S3 | Athena |
| PB-scale warehouse + joins + BI | Redshift (RA3 or Serverless) |
| Search + log analytics | OpenSearch Service |
| Cheap + reliable object store | S3 (class depends on access pattern) |
| ACID table on data lake | S3 Tables (Iceberg) |
| Airflow-native pipelines | Amazon MWAA |
| Serverless DAG with AWS-native retries | AWS Step Functions |
| Cron / event-driven fan-out | EventBridge + EventBridge Scheduler |
| SaaS → AWS connector | AWS AppFlow |
| DB → DB migration + CDC | AWS DMS (+ SCT for heterogeneous) |
| FGAC on data lake | AWS Lake Formation |
| Find PII in S3 | Amazon Macie |
| BI + natural-language data chat | Amazon QuickSuite (with Q in QuickSuite) |

Built by **CloudNinja** · **cloudninja.pro** · Free AWS practice exams & cheat sheets. No signup, no paywall.