Data Engineering Interview: Questions & Answers.

Data Engineering roles are in high demand globally, with hundreds of thousands of open positions. Employers consistently seek a mix of strong SQL and programming skills, big data pipeline experience, and the ability to design scalable systems. For example, an analysis of 2025 job postings found SQL mentioned in 61% of listings, Python in 56%, and cloud data warehouse or lake technologies in 78%. Core responsibilities include building reliable **ETL/ELT pipelines**, optimizing data storage (e.g. in **warehouses/lakehouses**), ensuring data quality, and collaborating across teams to deliver data for analytics and machine learning.

Cloud Data Stacks (AWS, Azure, GCP)

Focus: Key services on major cloud providers for data engineering – storage, compute, messaging, orchestration options, etc. Could include comparing AWS vs Azure vs GCP offerings (S3 vs ADLS vs GCS, Redshift vs Synapse vs BigQuery, Dataflow vs Glue vs Azure Data Factory, etc.). Also cost considerations and integration in cloud ecosystem.

Why It Matters: Many data engineering roles require working in one of the big clouds. Understanding the equivalent services and trade-offs indicates adaptability. Often asked to ensure candidate can design pipeline on X cloud, or compare with Y.

39. **(Easy)** What AWS services would you use to build a data lake and data warehouse?

Answer:

- For an AWS-based data lake, the core storage is typically **Amazon S3** (Simple Storage Service) as the data lake storage for raw and processed files (Parquet, Avro, etc.).
- You might use **AWS Glue** Data Catalog to catalog the data on S3 (so it's queryable with Hive/Spark), and AWS Glue ETL (which is basically managed Spark) or **AWS Lambda** or **AWS EMR** (Elastic MapReduce) for transformation jobs. Glue Catalog integrates with Athena, Redshift Spectrum, etc.
- For data warehouse, **Amazon Redshift** is AWS's cloud data warehouse (columnar MPP database for structured data)dice.com.
- Alternatively, AWS now also has **Redshift Spectrum** to query S3 data, bridging lake and warehouse.
- For real-time or messaging: **Amazon Kinesis** (or MSK – Managed Kafka) for streaming ingest.
- For orchestration: **AWS Data Pipeline** (older) or more commonly people run Airflow on AWS (Amazon MWAA - Managed Workflows for Apache Airflow) or use Step Functions for simple orchestration.
- For an end-to-end pipeline: For example, ingest data to S3 (maybe via Kinesis Firehose for streaming ingestion to S3), store long-term in S3 (lake). Use Glue or EMR to process into Parquet partitions. Then load into Redshift for fast BI queries. Glue Catalog holds table definitions so Athena (serverless SQL query on S3) or Redshift Spectrum can query lake data too.
- So summary answer: **S3** for lake storage, **Glue/Athena** for ETL and querying S3 data, **Redshift** for warehousing, and possibly **EMR or Glue jobs** for big data processing, plus other supporting services (Glue Catalog, Kinesis).
- (Azure answer would be ADLS, Databricks/Synapse, etc., GCP would be GCS + BigQuery + Dataflow, but question said AWS).

What interviewers look for: Knowledge of key AWS data services. S3 and Redshift are must-mention. Possibly expecting Athena or Glue as well since it's popular for serverless queries and managed Spark.

Common pitfalls:

- Confusing similar names (like saying "AWS BigQuery" – BigQuery is GCP, not AWS).
- Not mentioning S3 – if one says "HDFS on EC2" or something, that would seem outdated; AWS S3 is standard.
- Redshift vs Athena confusion: Athena is query engine on S3, Redshift is actual warehouse DB. It's good to mention both or at least Redshift for DW.

Follow-up: How does Redshift differ from Athena? (Answer: Redshift stores data internally in a cluster for low-latency queries, needs data loaded or via Spectrum; Athena is serverless querying directly on data in S3 with per-query cost, good for ad-hoc or low concurrency. Redshift better for high concurrency and performance on structured repetitive queries, at cost of cluster management.)

---

40. **(Easy)** Name an equivalent of Amazon S3 in Azure and GCP.

Answer:

- In Azure, the equivalent to S3 is **Azure Blob Storage** (part of Azure Storage Accounts) or specifically **Azure Data Lake Storage (ADLS) Gen2**, which is essentially blob storage with a hierarchical namespace optimized for analytics.
- In GCP (Google Cloud), the equivalent is **Google Cloud Storage (GCS)** – it's their object storage service, similar to S3 (with buckets, etc.).
- All three are massively scalable object stores for files/data lake use cases.
- Minor details: S3 uses buckets/objects with eventual consistency (though now strong consistency for new writes), GCS uses buckets similarly (with strong consistency by design). Azure Blob/ADLS has containers in storage accounts, integrated with Azure's AD for auth.
- But at high-level: S3 ~ GCS ~ Azure Blob.

What interviewers look for: Basic cloud knowledge to map services across providers. Likely expecting "Azure Blob Storage (Data Lake Storage) and Google Cloud Storage".

Common pitfalls:

- Naming Azure Data Lake without blob (should mention that ADLS Gen2 is built on Blob).
- Or mixing up something like thinking Azure "Data Lake" is separate service (older ADLS Gen1 was separate, Gen2 is integrated with Blob).
- On GCP, some might say "BigQuery" but BigQuery is a warehouse not object store.

Follow-up: How about an equivalent for AWS Lambda in Azure and GCP? (Not part of initial Q, but possible if they test cross-knowledge: Azure Functions, Google Cloud Functions as answers.)

---

41. **(Medium)** Compare AWS Redshift, Google BigQuery, and Azure Synapse Analytics at a high level.

Answer:

- **AWS Redshift:** A managed MPP data warehouse, where you provision a cluster of compute nodes. It uses columnar storage, requires you to load data (or query external via Spectrum). It's not fully serverless (though Redshift has an on-demand "Serverless" mode recently). You have to manage distribution keys, sort keys to optimize. Good for tight integration with AWS stack (S3 via Spectrum, uses AWS IAM, etc.). It bills by node-hours (or Redshift Serverless by usage).
- **Google BigQuery:** A serverless data warehouse. You don't provision nodes; you just run queries and it auto-scales using "slots" behind the scenesmedium.com. Storage and compute separated by default; you can query directly from GCS or native storage. It uses a SQL dialect and charges by bytes processed (for on-demand) or by reserved slot capacity. BigQuery handles a lot automatically (no indexing, no sort keys, it uses columnar Capacitor format and a huge underlying cluster).
- **Azure Synapse Analytics:** This encompasses what was Azure SQL Data Warehouse. It's an MPP warehouse as well. You provision data warehouse units (DWUs) or use the new Synapse workspace which can do on-demand serverless SQL on data lake too. It integrates with Azure Data Lake storage and Spark in Synapse workspace. Synapse tries to be an all-in-one analytics platform (with SQL engine, Spark engine, pipelines, etc.). The dedicated SQL pools are like Redshift concept (you allocate resources). Also has serverless SQL pool (similar to Athena concept for on-demand on files).
- **Performance & Scale:** All can handle terabytes to petabytes. BigQuery's separation makes it very flexible and often simpler to scale up/down quickly (just more slots). Redshift and Synapse dedicated pools you need to resize cluster for more power, which can take time (though Redshift RA3 nodes separate storage too).
- **Ecosystem integration:** If on AWS, Redshift integrates with AWS tools (Glue, S3, etc.). BigQuery integrates with GCP stack (Dataflow, etc., and built-in ML via BigQuery ML). Synapse integrates with Azure ML, Power BI etc. So often choice depends on cloud preference.
- **Cost:** BigQuery on-demand can be cost-effective for sporadic querying but expensive for heavy constant use (then reserved slots better). Redshift and Synapse you pay even if idle (unless you pause/resume in Synapse or use Redshift elastic resize).
- Also differences in SQL dialect and features: BigQuery standard SQL vs Redshift (Postgres-like SQL) vs Synapse (T-SQL based).
- So high-level: Redshift & Synapse are more alike (cluster-based), BigQuery is unique in being fully serverless by default.

What interviewers look for: Understanding of each and that they solve similar problem (cloud DW) but with different approaches (serverless vs cluster). Also if candidate can highlight one or two key differentiators like BigQuery's serverless nature or Synapse integration with Spark.

Common pitfalls:

- Calling Synapse just "SQL Data Warehouse" (that was old name, fine if mention though).

- Not knowing BigQuery's billing model (some think of it like a normal DB, but it's very different with per-query).
- Possibly forgetting Synapse has a serverless option (which is relatively new, but okay if focusing on dedicated SQL pool).

Follow-up: If your use case has highly variable workloads (some days heavy, some days light), which of these might be most cost-efficient? (Likely BigQuery on-demand, because you pay per use, whereas a static cluster might be underutilized on light days. Or Redshift Serverless now could also scale down.)

---

42. **(Medium)** How would you build a real-time analytics pipeline on Google Cloud?

Answer:

- For real-time ingest, use **Google Cloud Pub/Sub** (global message queue similar to Kafka) for capturing event streams.
- Stream processing: use **Google Cloud Dataflow**, which is GCP's managed Apache Beam runner, to consume from Pub/Sub, process events (window, aggregate, etc.) in near real-time. Dataflow can output to storage or BigQuery continuously.
- Alternatively, if the processing is simpler, one could use **BigQuery streaming inserts** directly, but usually Dataflow for any significant transformation.
- The pipeline might be Pub/Sub -> Dataflow (Beam) -> BigQuery (for storing aggregated results or raw events in table for analysis). BigQuery can handle streaming inserts (rows available within seconds for query).
- For serving to dashboard: BigQuery is directly queryable by tools like Google Data Studio, Looker, etc. If sub-second latency needed beyond BigQuery's ~seconds query, maybe push final aggregates to an in-memory store (like Cloud Memorystore / Redis) behind an API. But for many analytics, BigQuery's okay.
- Possibly use **Cloud Bigtable** or **Firestore** if we need NoSQL store for specific access patterns, but typical analytics would use BQ.
- Also mention Cloud Dataflow can do windowing, triggers for late data, etc. Exactly-once possible via Beam model (Pub/Sub + Dataflow ensures no dupes to sink).
- If using GCP fully managed alternatives: Could use **Cloud Functions** to do simple processing for each event (less common for heavy analytics though). Or **Dataproc** with Spark streaming (managed Hadoop/Spark cluster) but Dataflow is serverless and preferred for Beam.
- So recommended answer: *Pub/Sub as the ingestion buffer, Dataflow to process, BigQuery to store/query results in real-time.* Possibly mention enabling streaming buffer in BigQuery.

What interviewers look for: Knowledge of GCP data services and how to assemble them for streaming. They likely expect Pub/Sub & Dataflow (a common pairing) and BigQuery for analytics.

Common pitfalls:

- Suggesting outdated tech (like using Kafka on GCP instead of Pub/Sub – one could run Kafka, but GCP has Pub/Sub to avoid self-manage).
- Not including a stream processor – just saying Pub/Sub to BigQuery directly (you could use Pub/Sub subscriptions into BigQuery via Dataflow template though).
- Not understanding BigQuery streaming: BigQuery supports streaming inserts up to certain throughput, which Dataflow can do via BigQueryIO sink.

Follow-up: How does Dataflow ensure exactly-once processing in this pipeline? (This is deep, but likely via checkpointing and dedupe if needed in Beam, plus BigQuery sink is idempotent in that if Dataflow uses streaming insert, BigQuery could get duplicates if not careful, but Dataflow's BigQueryIO has best effort. Or using Pub/Sub message IDs to avoid reprocessing duplicates, etc.)

---

43. **(Hard)** How do you handle authentication/credentials when moving data between cloud services in a pipeline (e.g., reading from S3 in EMR, or writing to BigQuery from Dataflow)?

Answer:

- **Use cloud IAM roles and instance/service accounts** rather than embedding keys whenever possible:
  - In AWS: assign an IAM Role to the EMR EC2 instances that grants access to S3 (via instance profile). EMR nodes then automatically can access S3 without explicit creds. Similarly, Lambda or ECS tasks can assume roles.
  - In GCP: use a Service Account with appropriate OAuth scopes/IAM permissions. Dataflow jobs run with a service account (Dataflow's default SA or a custom one) which can be granted permission to BigQuery or Pub/Sub. No need for user to provide keys in code.
  - In Azure: use Managed Identities for resources (e.g., an Azure Data Factory or Databricks cluster can use a managed identity to access ADLS without secrets).
- If needing to handle credentials explicitly (not recommended in production): use secure storage of secrets (e.g., AWS Secrets Manager, GCP Secret Manager, Azure Key Vault) and retrieve them at runtime securely.
- For example, writing to BigQuery from Dataflow: ensure the Dataflow worker service account has BigQuery Data Editor permission. Then in pipeline code, you just use BigQueryIO – the system handles auth.

- Reading from S3 in Spark on EMR: the EMR EC2 Role has an S3 access policy, so Spark can read s3://... directly. Alternatively, use temporary STS credentials if cross-account, but generally role is easier.
- Emphasize principle of least privilege: grant only the needed access (e.g., read-only vs write).
- Also mention environment-specific injection: e.g., Airflow on AWS can use an IAM role via instance profile or connection with credential stored encrypted, etc.
- Summing: **use managed identities / roles** to avoid storing creds, or if needed store in a secret manager service.

What interviewers look for: Security awareness. That you don't say "hardcode keys" or something. Should mention IAM roles / service accounts.

Common pitfalls:

- Suggesting to just put access keys in code or config – immediate red flag.
- Not knowing the concept of service accounts or roles on at least one platform.
- Overcomplicating – the straightforward answer is use the cloud's identity system.

Follow-up: What if an on-premise system needs to push data to cloud storage – how to handle auth then? (Possible: use an access key but keep it in a secure vault on-prem, or set up a special service account with limited scope, or a gateway that handles auth. They want to see you consider secure secret management if identity federation not available.)

---

44. **(Hard)** Outline a data pipeline on Azure for a company migrating on-prem SQL data to Azure analytics services.

Answer:

- Likely components: **Azure Data Factory (ADF)** or **Azure Synapse Pipelines** to orchestrate and perform data movement from on-prem SQL Server to Azure.
- Use the **Self-hosted Integration Runtime** for ADF to securely connect to on-prem SQL and copy data out.
- The data could land into **Azure Data Lake Storage (ADLS) Gen2** in raw format (CSV, parquet). Then transform using **Azure Databricks (Spark)** or Synapse Spark to process data and perhaps load into **Azure Synapse Analytics (dedicated SQL pool)** or Azure SQL DB or **Azure Data Explorer** depending on use-case.
- Or, if near-real-time needed, use **Azure Data Factory mapping data flows** (which under the hood uses Spark) or **Azure Databricks** for transformation.

- Possibly use **Azure Data Factory** as overall orchestrator scheduling nightly copies, etc., with pipeline steps and monitors.
- For analytics, likely final store is **Azure Synapse Analytics** (which is basically Azure DW), or even directly use **Azure Analysis Services** or Power BI on top of Synapse for modeling.
- So, a pipeline:
  1. ADF pipeline triggers daily.
  2. Activity to Copy from on-prem SQL to ADLS (could be full extract or incremental using change tracking).
  3. Then an activity to run a Databricks Notebook or Synapse Spark job to clean/transform and save to a curated zone in ADLS or load to Synapse DW.
  4. Then possibly a step to refresh Power BI dataset or notify completion.
- Emphasize using Azure services: ADLS for storage, ADF for movement, Synapse/Databricks for processing, Synapse SQL for warehouse.
- Also mention **Azure Key Vault** for storing any credentials (like on-prem DB connection string).
- Possibly mention **Azure ExpressRoute or VPN** if large data to move and need stable connection, but probably too infra-level.

What interviewers look for: Cloud solution design ability – using right Azure tools. A structured approach (ingest, store, process, serve) hitting Azure tech at each stage.

Common pitfalls:

- Using wrong terms e.g., calling Synapse "Azure SQL DW" (old name, but okay if clarified).
- Not using ADF – ADF is often primary for migration pipelines.
- Not considering connectivity – self-hosted IR is important for on-prem to cloud copy.

Follow-up: How would you handle ongoing changes (CDC) from on-prem SQL to keep Azure in sync? (One might use Azure Data Factory with incremental copy via a watermark, or use on-prem SQL CDC to capture changes and push, or use Azure Data Factory's built-in change data capture if available. Or use Debezium and Kafka, but staying within Azure, maybe use Azure Event Hubs and Change Feed.)

Sources (Cloud Stacks):

- **Dice article** – lists cloud platform skills (just general)dice.com
- **BMC Blog** – AWS vs Azure vs GCP comparison (for general differences)bmc.com
- **Hevo or Panoply** – blogs comparing Redshift, BigQuery, Snowflake, Synapse, etc., likely similar content to what's givenmedium.com
- **Google Cloud docs** – BigQuery architecture highlights (serverless, separation)panoply.iopanoply.io
- **Azure docs** – mention ADLS Gen2 as HDFS-compatible store
- **GCP guides** – real-time architecture (Pub/Sub + Dataflow + BigQuery) guidance, maybe in solutions or blog
- **AWS docs** – typical reference architectures (e.g., Kinesis + EMR + Redshift etc.), but not specifically scraped above