Question: Please describe specific work project experience from your current and recent work, where you contributed at least 50% of the effort.
Answer:
As a staff engineer at Legend Energy Advisors, I was tasked with finding and building a solution for a centralized observability platform that would handle log aggregation, different types of health checks, tracing, displaying runtime data, incident monitoring and analysis, suggesting best actions taken on specific event within the system. I have owned and built a platform that accomplishes all this. I used Go for the API, agent and middleware logic, Angular and D3 were used for the frontend, Logstash for log shipper, ElastcSearch for log storage. I used CQRS pattern and Event Sourcing for efficiency and consistency of architecture. I also built a Terraform/Ansible CI/CD process. I used, Logstash as log shipper, ElastcSearch for storage, Apache Kafka and AWS EMR for streaming. I have also implemented Terraform/Ansible CI/CID for the platform. The platform was deployed to AWS and used for observability of tens of EC2 instances on ELB cluster, as well as onprem servers within a Proxmox Debian cluster. It worked on terabytes of data daily. This was one of my most impactful projects and it was extremely helpful for streamlining company's observability needs and providing better process for incident analysis and response.

<u>AWS Proficiency</u>

Question: Can you describe your experience with AWS services such as S3, Redshift, and EMR?
Answer:
I have over 12 years of experience working with different AWS services, including extensive use of S3, Redshift, and EMR.
I have extensively used S3 as storage layer for APIs, both standalone and exposed via API Gateway, and multiple Lambda Functions deployed on AWS.
I've have also used S3 as the central storage solution for large-scale data lakes, implementing best practices like versioning, lifecycle policies, and data partitioning to optimize performance and cost.

With EMR, I've managed distributed data processing for tasks like log analysis, observability and real-time streaming pipelines, leveraging Apache Spark for scalability and efficiency.

As for Redshift, and Snowflake, I've designed and optimized data warehouse architectures, implementing distribution keys, sort keys, and compression to support high-performance analytical workloads. This experience has given me deep expertise in building scalable, cost-effective, and reliable AWS-based architectures.

Question: How have you optimized data storage and retrieval in AWS environments?
Answer:
I optimized data storage and retrieval in AWS by choosing the proper storage solution based on performance, scalability, and cost. For high-volume, structured data, I use Amazon Redshift, optimizing queries through proper distribution keys, compression, and partitioning. For unstructured or semi-structured data, I leverage Amazon S3 with lifecycle policies and data partitioning, and I use AWS Glue for cataloging. I also employ Amazon Athena for serverless querying of S3 data and Amazon Elasticsearch for full-text search capabilities. For real-time data, I use Kinesis or Amazon MSK to stream data efficiently into the processing pipeline.

Additionally, I implement caching strategies with Amazon ElastiCache and optimize read/write operations through data locality and indexing techniques.

Medallion Delta Lake

Question: What is your understanding of the Medallion architecture in Delta Lake, and how have you implemented it in past projects?

Answer:

The Medallion architecture in Delta Lake is a layered approach to data organization that enhances data quality and scalability. It consists of three layers: Bronze, for raw, unprocessed data ingested from multiple sources; Silver, for cleansed and enriched data with applied transformations; and Gold, for business-ready, aggregated datasets optimized for analytics.

In past projects, I implemented this architecture by using Delta Lake on Apache Spark for its ACID transactions and schema enforcement. I ingested data into the Bronze layer from streaming sources like Kafka or batch systems via Apache Spark, applied deduplication and validation in the Silver layer, and created aggregations or curated datasets in the Gold layer for BI tools like Tableau. Automated pipelines, built with orchestration tools like Airflow, ensured seamless data flow between layers, while Delta's time travel feature allowed for auditability and debugging.

Question: Can you discuss a specific project where you used Delta Lake to manage large-scale data?

Answer (including Architecture Overview, Pipeline Desing and Results Summary):

In a recent project for a large e-commerce platform, I was responsible for designing and implementing a scalable data pipeline to handle large-scale data processing using Delta Lake on AWS. The goal was to consolidate data from multiple sources, perform ETL transformations, and prepare it for analytics in near real-time. The project required handling both batch and streaming data at scale, with a focus on ensuring data quality, consistency, and performance.

Architecture Overview:

I used AWS as our primary cloud platform, utilizing several AWS services to build a robust, scalable solution:

Amazon S3 was used as the central storage layer for all raw and processed data.
Apache Spark on Amazon EMR was used for distributed data processing.
Delta Lake was employed on top of Spark to manage our data in a highly reliable, ACID-compliant, and scalable manner.
Apache Airflow was used for orchestrating and scheduling the ETL workflows.
AWS Glue provided the data cataloging and schema management services.

Pipeline Design and Implementation:

1. Ingestion (Bronze Layer): We ingested data from various sources, including transactional databases, clickstream logs, and external APIs, into Amazon S3 in its raw form. This raw data was stored in a Bronze layer in Delta Lake tables. To handle streaming data, I used Kinesis Data Streams for real-time ingestion and then consumed the streams with Apache Spark to load the data into S3. The raw data in S3 was partitioned by date and other relevant keys, ensuring efficient storage and retrieval.
Apache Spark was configured to process both batch and streaming data simultaneously.
For batch data, we scheduled daily ingestion jobs using Airflow, while Kinesis handled the real-time streaming data.

2. Transformation and Enrichment (Silver Layer): After the initial ingestion, data required cleansing and enrichment. In the Silver layer, we performed a series of transformations to clean and enrich the data, including:

2.1. Deduplication
2.2. Standardizing data formats
2.3. Joining with external data sources for enrichment (e.g., customer data, product information)

For this, I used Spark with Delta Lake's ACID transaction support, which ensured consistency during the transformation process even when multiple jobs were running in parallel. I also used Delta Lake's schema evolution feature to handle evolving data structures, which is critical when working with semi-structured data sources.

3. Aggregation and Analysis (Gold Layer): Once the data was cleaned and enriched, we moved it to the Gold layer for final aggregation and analytics. This layer contained pre-aggregated data for reporting and business intelligence purposes. For example, I aggregated clickstream data into session-based metrics and merged them with transactional data to provide business insights.
Data in the Gold layer was partitioned by key dimensions such as product categories and customer segments to optimize query performance. I used AWS Glue to catalog the Delta tables, making them easily accessible for BI tools like Amazon QuickSight.

4. Orchestration with Apache Airflow: To manage the entire pipeline, I used Apache Airflow for scheduling and orchestrating the ETL jobs. Airflow DAGs were used to manage dependencies between tasks such as:

Data ingestion from various sources
Transformation and data quality checks
Aggregation and writing to the Gold layer

Airflow's flexibility allowed us to add custom logic for retries and error handling, and it provided visibility into job statuses through its UI, making the pipeline easier to monitor and manage. For complex workflows, we also integrated Airflow with AWS Step Functions to manage multi-step processes.

5. Data Quality and Consistency: Delta Lake played a key role in ensuring data quality and consistency across all stages of the pipeline. Its ACID properties allowed us to:

Ensure that only valid and fully processed data was available to downstream consumers.
Handle schema changes seamlessly, which was critical in this project due to the evolving nature of the data.
Perform time travel to access previous versions of the data for debugging and auditing purposes.

Additionally, we implemented data validation and monitoring mechanisms using AWS CloudWatch to track metrics like data volume and processing times. Alerts were set up for anomaly detection, ensuring that any data issues were caught early in the pipeline.

Results and the Impact:

The solution allowed to:

1. Handle terabytes of data on a daily basis with low latency, even with both batch and streaming data.
2. Achieve high data consistency and integrity thanks to Delta Lake's ACID transactions.

3. Provide near real-time analytics capabilities, which were critical for decision-making in marketing and sales.
4. Enable faster and more efficient querying through data partitioning and indexing, making it easier for analysts and business users to get insights from the data.
5. By combining Delta Lake with Spark and AWS services like Kinesis and Airflow, we were able to build a robust, scalable, and highly efficient data pipeline. This architecture not only met the project's requirements for performance and scalability but also provided flexibility for future data sources and evolving business needs."

Security

Question: How do you ensure data security and compliance in cloud environments, particularly with AWS?

Answer:
In AWS, I ensure data security and compliance by implementing a layered approach. This includes encrypting data at rest and in transit using AWS-managed or customer-managed keys, enforcing least-privilege access through IAM roles and policies, and using services like AWS Config and CloudTrail for continuous monitoring and auditing. I also leverage security groups, VPCs, and private endpoints for network-level security. For compliance, I align configurations with frameworks like SOC 2, PCI or HIPAA, I also use AWS Artifact for compliance documentation, and implement guardrails with AWS Control Tower or Service Control Policies. I use properly defined IAM roles, groups, users and policies to manage access.

Question: Can you describe a time when you had to address a significant security vulnerability in a data platform?

Answer:
While I've been at my Staff Software Engineer role at Legend Energy Advisors, we discovered a critical security vulnerability in one of our customer's cloud environments: an S3 bucket with misconfigured IAM access that became publicly accessible. I immediately assessed the scope of the exposure, identified the affected data, and secured the bucket by implementing appropriate access policies and enabling bucket-level encryption. To prevent future incidents, I introduced automated compliance checks with AWS Config and set up alerts through CloudWatch for policy violations. Additionally, I conducted a root-cause analysis and trained the team on secure cloud practices, IAM role managemenet and defining proper acess scoping. The incident reinforced the importance of continuous monitoring and proactive security audits in our environments. It helped the organization to gain more awareness of vulnerabilities and the ways to handle them looking forward.

ETL Processes

Question: What strategies do you use for efficient ETL processes in a cloud environment?
Answer:
I designed efficient ETL processes by aligning them with the specific type of architecture, data being structured or unstructured — data lakes or warehouses — depending on the use case. For data lakes, I optimize storage and retrieval using partitioning, formats like Parquet or ORC, and frameworks like Apache Spark for distributed processing. In data warehouses, I use ELT strategies, leveraging the computational power of systems like Snowflake or Redshift for in-

place transformations. For orchestration, I use tools like Apache Airflow, Dagster, or Luigi to define DAGs for modular, reusable workflows. These tools enable dependency management, retries, and scalability. This approach ensures efficient processing, cost optimization, and maintainability in the cloud.

Question: How do you handle data quality and integrity during ETL operations?
Answer:
I ensure data quality and integrity in ETL processes by implementing validation and transformation steps at multiple stages. In AWS, I use AWS Glue DataBrew for data profiling and cleansing, I also use Lambda functions for custom validation rules, and I use Glue ETL jobs to enforce schema consistency. I leverage AWS DMS for real-time data replication with built-in validation, and I enable constraints like primary keys and referential integrity in Redshift for downstream quality checks. Additionally, I use AWS Lake Formation for fine-grained access control and CloudWatch for monitoring data pipelines, ensuring any anomalies are quickly identified and resolved.

Real-Time Data Processing

Question: Can you explain your experience with real-time data processing and the tools you've used?
Answer:
As a software architect, data archietct, software engineer and data engineer have extensive experience with real-time data processing using both Apache Kafka and AWS Kinesis. With Apache Kafka, I've implemented high-throughput, low-latency real-time pipelines for event streaming, leveraging Kafka Connect for integration and schema management with Confluent Schema Registry. When I worked in AWS, I used Amazon MSK for managed Kafka clusters, ensuring scalability and durability. Additionally, I work with Amazon Kinesis Data Streams for real-time ingestion, Kinesis Data Firehose for delivery to S3 or Redshift, and Kinesis Data Analytics for stream processing with SQL. I integrated these systems with AWS Lambda for event-driven processing and use CloudWatch for end-to-end monitoring and alerting. These tools enabled me to deliver reliable, scalable streaming solutions.

Question: How do you ensure low latency and high throughput in real-time data pipelines?
Answer:
I ensure low latency and high throughput in real-time data pipelines by leveraging best practices for both Apache Kafka and AWS Kinesis. With Kafka, I optimize producer and consumer configurations, such as batching, compression, and acknowledgment settings, while ensuring efficient partitioning and replication. I also monitor throughput and latency using Kafka metrics and auto-scale partitions or brokers as needed.

In AWS Kinesis, I use proper shard allocation and partition keys for balanced data distribution, while integrating Kinesis Data Analytics or Apache Flink for low-latency processing. For both platforms, I implement VPC endpoints to reduce network overhead and CloudWatch or Prometheus for monitoring. Finally, I use horizontal scaling and resource optimization to maintain consistent performance even under high load.