Day 10: Advanced Azure Databricks

Welcome to Day 10 of our Azure Data Engineer interview questions and answers series! Today, we will focus on advanced topics related to Azure Databricks. These questions will challenge you to delve deeper into the capabilities and optimization strategies of Databricks, helping you prepare for more complex interview scenarios.

1. What is Databricks Delta and how does it enhance the capabilities of Azure Databricks?

- **Answer:** Databricks Delta, now known as Delta Lake, is an open-source storage layer that brings ACID transactions to Apache Spark and big data workloads. It enhances Azure Databricks by providing features like:
    1. ACID transactions for data reliability and consistency.
    2. Scalable metadata handling for large tables.
    3. Time travel for data versioning and historical data analysis.
    4. Schema enforcement and evolution.
    5. Improved performance with data skipping and Z-ordering.

2. Explain how you can use Databricks to implement a Medallion Architecture (Bronze, Silver, Gold).

- **Answer:**
    1. **Bronze Layer (Raw Data):** Ingest raw data from various sources into the Bronze layer. This data is stored as-is, without any transformation.
    2. **Silver Layer (Cleaned Data):** Clean and enrich the data from the Bronze layer. Apply transformations, data cleansing, and filtering to create more refined datasets.
    3. **Gold Layer (Aggregated Data):** Aggregate and further transform the data from the Silver layer to create high-level business tables or machine learning features. This layer is used for analytics and reporting.

3. How can you use Azure Databricks for real-time data processing?

- **Answer:**
    1. Use Azure Event Hubs or Azure IoT Hub to ingest real-time data streams.
    2. Create a Databricks Structured Streaming job to process the streaming data.
    3. Perform transformations and aggregations on the streaming data using Spark SQL or DataFrame API.
    4. Output the processed data to a storage service like ADLS, Azure SQL Database, or a real-time dashboard.

4. Describe the role of MLflow in Azure Databricks and how it helps in managing the machine learning lifecycle.

- **Answer:** MLflow is an open-source platform for managing the end-to-end machine learning lifecycle. In Azure Databricks, MLflow helps by providing:
    1. **Experiment Tracking:** Log parameters, metrics, and artifacts from ML experiments to track performance and reproducibility.

2. **Model Management:** Register, version, and organize models in a centralized model registry.
3. **Deployment:** Deploy models to various environments, including Databricks, Azure ML, and other platforms.
4. **Reproducibility:** Ensure experiments are reproducible with tracked code, data, and configurations.

5. What is AutoML in Azure Databricks, and how can it simplify the machine learning process?

- **Answer:** AutoML in Azure Databricks automates the process of training and tuning machine learning models. It simplifies the machine learning process by:
  1. Automatically selecting the best model algorithm based on the data.
  2. Performing hyperparameter tuning to optimize model performance.
  3. Providing easy-to-understand summaries and visualizations of model performance.
  4. Allowing data scientists and engineers to focus on higher-level tasks instead of manual model selection and tuning.

6. Scenario: You need to implement a data governance strategy in Azure Databricks. What steps would you take?

- **Answer:**
  1. **Data Classification:** Classify data based on sensitivity and compliance requirements.
  2. **Access Controls:** Implement role-based access control (RBAC) using Azure Active Directory.
  3. **Data Lineage:** Use tools like Databricks Lineage to track data transformations and movement.
  4. **Audit Logs:** Enable and monitor audit logs to track access and changes to data.
  5. **Compliance Policies:** Implement Azure Policies and Azure Purview for data governance and compliance monitoring.

7. Scenario: You need to optimize a Spark job that has a large number of shuffle operations causing performance issues. What techniques would you use?

- **Answer:**
  1. **Repartitioning:** Repartition the data to balance the workload across nodes and reduce skew.
  2. **Broadcast Joins:** Use broadcast joins for small datasets to avoid shuffle operations.
  3. **Caching:** Cache intermediate results to reduce the need for recomputation.
  4. **Shuffle Partitions:** Increase the number of shuffle partitions to distribute the workload more evenly.
  5. **Skew Handling:** Identify and handle skewed data by adding salt keys or custom partitioning strategies.

8. Scenario: You are working with a large dataset that requires frequent schema changes. How would you handle schema evolution in Delta Lake?

- **Answer:**
    1. Enable Delta Lake's schema evolution feature by setting `mergeSchema` to `true` when writing data.
    2. Use `ALTER TABLE` statements to manually update the schema if necessary.
    3. Implement a versioning strategy using Delta Lake's time travel feature to keep track of schema changes over time.
    4. Monitor and validate schema changes to ensure they do not break downstream processes or analytics.

9. How would you secure and manage secrets in Azure Databricks when connecting to external data sources?

- **Answer:**
    1. Use Azure Key Vault to store and manage secrets securely.
    2. Integrate Azure Key Vault with Azure Databricks using Databricks-backed or Azure-backed scopes.
    3. Access secrets in notebooks and jobs using the `dbutils.secrets` API.
    4. Ensure that secret access policies are strictly controlled and audited.

10. Scenario: You need to migrate an on-premises Hadoop workload to Azure Databricks. Describe your migration strategy.

- **Answer:**
    1. **Assessment:** Evaluate the existing Hadoop workloads and identify components to be migrated.
    2. **Data Transfer:** Use Azure Data Factory or Azure Databricks to transfer data from on-premises HDFS to ADLS.
    3. **Code Migration:** Convert Hadoop jobs (e.g., MapReduce, Hive) to Spark jobs and test them in Databricks.
    4. **Optimization:** Optimize the Spark jobs for performance and cost-efficiency.
    5. **Validation:** Validate the migrated workloads to ensure they produce the same results as on-premises.
    6. **Deployment:** Deploy the migrated workloads to production and monitor their performance.