**Basic Level Questions**

**1. What is Azure Data Factory (ADF)?**

* Azure Data Factory (ADF) is a cloud-based ETL and data integration service that allows you to create data-driven workflows for orchestrating and automating data movement and data transformation.

**2. What are the key components of Azure Data Factory?**

* **Pipelines** – Define workflows to perform data movement and transformation.
* **Activities** – The steps performed within a pipeline (e.g., copy, lookup).
* **Datasets** – Data structures that represent the data you want to work with.
* **Linked Services** – Connection strings to external data sources.
* **Integration Runtime (IR)** – Compute infrastructure for data movement and transformation.

**3. What is a Linked Service in Azure Data Factory?**

* A Linked Service defines the connection to data sources and computer environments (e.g., Blob Storage, SQL Database, Redshift).

**4. What types of Integration Runtimes are available in ADF?**

* **Azure IR** – For cloud-based data movement and transformation.
* **Self-Hosted IR** – For on-premises data movement.
* **SSIS IR** – For running SSIS packages.

**5. How does ADF handle data movement?**

* Data movement is handled using **Copy Activity** within pipelines.
* ADF supports moving data between over **90 connectors** (Azure Blob Storage, SQL Server, AWS S3, Redshift, etc.).

**Intermediate Level Questions**

**6. What is the difference between Pipeline and Activity in ADF?**

* **Pipeline** – A logical container for a set of activities.
* **Activity** – A single step or task within a pipeline (e.g., Copy, Execute SSIS).

**7. What are the types of triggers in ADF?**

* **Schedule Trigger** – Runs at a specific time.
* **Tumbling Window Trigger** – Runs at a fixed interval.
* **Event-Based Trigger** – Fires when a blob is created or deleted.

**8. How can you secure data in Azure Data Factory?**

* Managed Identity for authentication.
* Data encryption using Azure Key Vault.
* Role-based Access Control (RBAC).

**9. How to handle failures in ADF pipelines?**

* Use **Try-Catch logic** using pipeline activities.
* Configure **Retry Policy** on activities.
* Use **Alerting and Monitoring** through Azure Monitor.

**10. How does ADF manage data partitioning for large data sets?**

* **Data Partitioning** options are available in **Copy Activity**.
* Types:
  + Dynamic Range Partitioning
  + Fixed Size Partitioning

**Advanced Level Questions**

**11. What are the different debugging options in Azure Data Factory?**

* **Data Preview** – To preview data in Data Flows.
* **Pipeline Debug** – To test pipeline execution without publishing.
* **Monitoring** – Track pipeline runs and activity failures.

**12. What is Mapping Data Flow in ADF?**

* A **low-code, visual data transformation** feature.
* Allows ETL/ELT transformations using a drag-and-drop interface.

**13. How do you implement CI/CD (Continuous Integration/Deployment) in ADF?**

* Use **Git Integration** with Azure DevOps or GitHub.
* Use **ARM templates** for deployment.
* Use **Release Pipelines** to automate deployment.

**14. How does ADF integrate with Databricks?**

* Use **Databricks Activity** within pipelines.
* Can pass parameters and handle Databricks notebooks.

**15. What are the limitations of Azure Data Factory?**

* No built-in data catalog.
* No real-time data processing (use Azure Stream Analytics for that).
* Limited row-level transformations (handled better by Databricks).

**16. How do you monitor pipeline runs in ADF?**

* Azure Monitor
* Application Insights
* Log Analytics

**17. What is the difference between Mapping Data Flow and Wrangling Data Flow?**

| **Mapping Data Flow** | **Wrangling Data Flow** |
| --- | --- |
| Based on Spark engine | Based on Power Query engine |
| Handles big data transformations | Designed for self-service data prep |
| Low-code | No-code interface |

**18. How can you parameterize a pipeline in ADF?**

* Define pipeline parameters at the start.
* Use expressions like @pipeline().parameters.paramName.

**19. How do you handle incremental data loading in ADF?**

* Use **Watermark columns** to track last-loaded records.
* Use **Lookup Activity** and **Copy Activity** to load only new data.

**20. What is a ForEach activity in ADF?**

* A loop activity that allows you to process multiple items in parallel or sequentially.

**⭐ Scenario-Based Questions**

**21. How would you design a pipeline to load data from an SFTP to Azure SQL DB?**

1. Create Linked Services for SFTP and Azure SQL DB.
2. Create a dataset for SFTP and SQL DB.
3. Use **Copy Activity** to load data.
4. Use **Lookup Activity** to handle incremental load.
5. Add error handling using **Try-Catch**.

**22. How would you handle schema drift in ADF?**

* Enable **Schema Drift** in Mapping Data Flow.
* Define default handling strategies.
* Use **Alter Row** to handle schema changes.

**23. How do you handle large file processing in ADF?**

* Use **Parallel Processing** in Copy Activity.
* Split data into chunks using dynamic partitioning.
* Use **Batch processing** in Data Flows.

**24. How would you implement error logging in ADF?**

* Create a separate table for error logging.
* Capture errors using **Set Variable** and **Web Activity**.
* Send alerts using **Azure Monitor**.

**25. How would you move data from on-premises to Azure using ADF?**

1. Install **Self-Hosted Integration Runtime** on-premises.
2. Create a Linked Service for on-premises DB.
3. Use **Copy Activity** to move data.
4. Monitor using **Azure Monitor**.

**1. Optimize Data Movement**

* **Use Staging:**
  + When copying large datasets, enable staging in **Azure Blob** or **ADLS** to increase throughput.
  + Staging allows parallel reads and writes, boosting performance.
* **Parallel Copying:**
  + Set the **degree of parallelism** to maximum (based on source and destination capacity).
  + Split large datasets into smaller partitions to process them concurrently.
* **Use Appropriate Integration Runtimes:**
  + Use **Self-Hosted Integration Runtimes** for on-premises data.
  + Use **Azure Integration Runtimes** for cloud-based data.
  + Scale up the integration runtime nodes if needed.

**2. Optimize Data Flow Performance**

* **Set Appropriate Data Flow Settings:**
  + Increase **compute type** (general-purpose vs. memory-optimized).
  + Scale out using the **number of cores** and nodes to handle large volumes of data.
* **Reduce Data Shuffling:**
  + Use **broadcast joins** instead of shuffle joins where possible.
  + Pre-sort data to reduce repartitioning.
* **Use Cached Lookup:**
  + For repeated lookup operations, cache data instead of reloading.
* **Optimize Memory Usage:**
  + Use only the necessary columns and rows.
  + Disable **unnecessary transformations**.

**3. Optimize Pipeline Performance**

* **Enable Pipeline Concurrency:**
  + Set the **concurrency limit** to allow multiple pipeline runs in parallel.
* **Use Partitioning:**
  + When moving data from sources like SQL or Blob, enable partitioning to distribute the load.
  + Example: Partition large files by date or key column.
* **Reduce Unnecessary Activities:**
  + Avoid sequential dependencies where parallel execution is possible.
  + Combine smaller pipelines into one if they can run in parallel.

**4. Optimize Linked Services and Datasets**

* **Set Efficient Timeout and Retry Policy:**
  + Reduce timeout for quick-failing sources.
  + Set intelligent retry policies based on error types.
* **Use Managed Identity for Authentication:**
  + Managed identities offer better connection handling and security.

**5. Monitor and Debug Performance Issues**

* **Enable Logging:**
  + Use **Activity Runs** and **Pipeline Runs** to monitor duration and bottlenecks.
  + Enable **Data Flow Debug** mode for deeper analysis.
* **Check Azure Monitor and Log Analytics:**
  + Review integration runtime logs.
  + Use Azure Monitor to identify long-running or failing activities.

**✅ Best Practices:**

✔️ Minimize data movement across regions.  
✔️ Keep activity and pipeline structures simple.  
✔️ Limit data flow memory usage to avoid auto-scaling delays.  
✔️ Test performance regularly using different configurations.

**🔥 1. Data Movement Activities**

Activities used to move data between different sources and destinations.

**✅ Copy Data**

* Moves data between supported data stores (SQL, Blob, ADLS, etc.).
* Supports parallelism, compression, and staging.
* Can handle large volumes of data.

**✅ Data Flow**

* Processes and transforms data in a scalable way using Spark.
* Allows operations like joins, aggregations, and lookups.

**🚀 2. Data Transformation Activities**

Activities used to clean, format, and transform data.

**✅ Stored Procedure**

* Executes a stored procedure on a SQL database.
* Useful for ETL post-processing or validation.

**✅ Lookup**

* Retrieves a single row or value from a dataset.
* Often used to pass dynamic parameters.

**✅ Script**

* Runs a SQL or other script directly on a target database.

**✅ Data Flow**

* Allows complex transformations like:
  + Aggregation
  + Sorting
  + Pivoting
  + Join and Union

**🔄 3. Control Flow Activities**

Activities that control the flow of execution.

**✅ ForEach**

* Iterates over a collection of data and processes each item individually.
* Supports parallel execution.

**✅ Until**

* Executes activities repeatedly until a specific condition is met.

**✅ If Condition**

* Executes a set of activities based on a true/false condition.
* Similar to an "IF-ELSE" statement in programming.

**✅ Switch**

* Routes execution based on the value of an expression (like a CASE statement).

**📡 4. External Activities**

Activities used to integrate with external systems or services.

**✅ HDInsight**

* Runs Hadoop, Spark, or Hive jobs on an HDInsight cluster.

**✅ Databricks Notebook**

* Triggers a Databricks notebook and passes dynamic parameters.
* Supports scalable data processing.

**✅ Machine Learning**

* Calls an Azure Machine Learning pipeline or model.

**🎯 5. Azure-Specific Activities**

Activities to integrate with other Azure services.

**✅ Azure Function**

* Triggers an Azure Function for custom logic or processing.

**✅ Web**

* Makes HTTP requests to REST endpoints.
* Often used to trigger other services or APIs.

**✅ Azure Batch**

* Executes batch jobs for high-performance parallel processing.

**📅 6. Scheduling and Monitoring Activities**

Activities used to manage execution and monitor processes.

**✅ Wait**

* Pauses the pipeline execution for a specified time.

**✅ Set Variable**

* Assigns a value to a pipeline variable.

**✅ Get Metadata**

* Retrieves metadata from a dataset (e.g., file size, last modified date).

**✅ Validation**

* Validates if a file or table exists before proceeding.

**🔐 7. Security and Notification Activities**

Activities for handling security and notifications.

**✅ Send Email**

* Sends email notifications based on success/failure.

**✅ Azure Key Vault**

* Retrieves secure secrets (e.g., passwords, connection strings).

**✅ Example Use Case: ETL Pipeline**

1. Use **Copy Data** to move raw data from Blob to SQL.
2. Use a **Data Flow** to clean and transform the data.
3. Use a **Stored Procedure** to load the data into the final table.
4. Use **If Condition** to check for errors and send an **Email** if any issues arise.

**How to Implement SCD in ADF:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SCD Type** | **Description** | **When to Use** | **Complexity** |
| **Type 1** | **Overwrite existing record** | **No need to track history** | **Low** |
| **Type 2** | **Insert new record and maintain history** | **Track full history of changes** | **Medium** |
| **Type 3** | **Keep previous and current value** | **Track only recent change** | **Low** |

**. SCD Type 1 – Overwrite Existing Data**

**Steps:**

1. Create a **Copy Data Activity** to load source data into a staging table.
2. Use a **Data Flow** to compare the staging table with the target table based on a unique key.
3. Use a **Merge Operation** in Data Flow:
   * If a record exists → Overwrite with new values.
   * If a record doesn’t exist → Insert as a new row.
4. Map the output to the target table.

**Example Data Flow:**

* **Source → Lookup on Target Table → Conditional Split → Sink**
* Match condition: source.CustomerID = target.CustomerID
* **Update Logic:** Overwrite existing record
* **Insert Logic:** Insert new record

**2. SCD Type 2 – Maintain History**

**🛠️ Steps:**

1. Create a **Copy Data Activity** to load source data into a staging table.
2. Use a **Data Flow** for comparison and versioning:
   * Compare source and target records using a unique key.
   * If a match is found, check if any value has changed.
3. In the **Data Flow**:
   * If values have changed →
     + Mark the existing record as **Inactive** (set EndDate to current date).
     + Insert a new row with updated data and set StartDate to the current date.
   * If no match → Insert new row with StartDate

**Example Data Flow:**

* **Source → Lookup on Target Table → Conditional Split → Sink**
* **Match condition:** source.CustomerID = target.CustomerID
* **Condition:**
  + If record found and changed → Mark EndDate and Insert New Row
  + If record not found → Insert New Row