🔥 Most Important Azure Data Factory (ADF) Interview Questions & Answers 🔥

If you're preparing for an **Azure Data Factory (ADF) interview**, here are some **commonly asked questions** along with their **answers** to help you succeed! 🚀

1. What is Azure Data Factory (ADF)?

💡 **Answer:**
Azure Data Factory is a **cloud-based ETL (Extract, Transform, Load) and data integration service** used to **orchestrate and automate data workflows** across different sources. It helps in **moving, transforming, and loading data** into cloud-based storage and analytics solutions like **Azure Data Lake, Azure SQL, and Synapse Analytics**.

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

💡 **Answer:**
ADF consists of the following key components:

✅ **Pipelines** – Group of activities that perform data movement & transformation
✅ **Activities** – Steps in a pipeline (e.g., Copy, Data Flow, Stored Procedure)
✅ **Datasets** – Define the structure of data sources/destinations
✅ **Linked Services** – Connections to external data stores (e.g., ADLS, SQL, APIs)
✅ **Integration Runtime (IR)** – Compute engine to execute activities
✅ **Triggers** – Schedule or event-driven execution of pipelines

3. What are the different types of Integration Runtimes (IR) in ADF?

💡 **Answer:**
Azure Data Factory supports **three types** of Integration Runtimes:

🔹 **Azure IR** – Used for cloud-based data movement & transformation

🔹 **Self-Hosted IR** – Used for **on-prem** or **private network** data access
🔹 **Azure-SSIS IR** – Used to **run SSIS packages in the cloud**

4. What are the different types of triggers in ADF?

💡 **Answer:**
ADF provides **three types** of triggers:
1️⃣ **Schedule Trigger** – Runs pipelines on a specific time-based schedule
2️⃣ **Tumbling Window Trigger** – Runs pipelines at fixed intervals with dependency tracking
3️⃣ **Event-Based Trigger** – Runs pipelines when a file is added/deleted in **ADLS/Blob Storage**

5. How do you move data from On-Prem to Azure using ADF?

💡 **Answer:**
To move data from an **on-premises** database to **Azure**, follow these steps:
✅ **Install Self-Hosted Integration Runtime (SHIR)** on an on-prem server
✅ **Create a Linked Service** in ADF to connect to the on-prem database
✅ **Use Copy Data Activity** to move data to Azure (ADLS, Blob, SQL, Synapse)
✅ **Schedule pipeline execution** using triggers

6. How does Azure Data Factory handle failures?

💡 **Answer:**
ADF provides several **error-handling mechanisms**:
✔ **Retry Policy** – Set retry count & interval for transient failures
✔ **Logging & Monitoring** – Use **Azure Monitor & Log Analytics**
✔ **Custom Error Handling** – Implement **If Condition & Web Activity** for alerts
✔ **Try-Catch Logic** – Use **Execute Pipeline & Error Handling**

7. What are Mapping Data Flows in ADF?

💡 **Answer:**
Mapping Data Flows provide **a no-code, visual way** to perform **ETL transformations** using **Apache Spark** in ADF. Features include:

✅ **Joins, aggregations, filtering, pivoting**
✅ **Schema drift support** (Handles changing schema)
✅ **Auto-scaled Spark execution**

📌 **Example Use Case:** Cleansing & transforming raw data before loading it into **Azure Synapse Analytics**.

8. What is the difference between ADF and SSIS?

💡 **Answer:**

Feature	Azure Data Factory (ADF)	SQL Server Integration Services (SSIS)
Deployment	Cloud-based (PaaS)	On-Premises (IaaS)
Data Movement	Works with hybrid & cloud sources	Limited to SQL-based environments
Scalability	Auto-scales	Fixed resource allocation
Execution Mode	Serverless	Requires VM setup
Orchestration	Advanced workflow automation	Limited control flow

🔷 **ADF is a modern, scalable, cloud-native alternative to SSIS for hybrid data movement & orchestration.**

9. How can you secure data movement in ADF?

💡 **Answer:**
✓ **Use Managed Identity Authentication** instead of storing credentials
✓ **Encrypt data at rest & in transit** (TLS 1.2, Azure Key Vault)
✓ **Use Private Endpoints** to restrict data access
✓ **Monitor data pipelines using Azure Security Center**

10. What is the difference between Copy Data Activity and Data Flow?

💡 **Answer:**

Feature	Copy Data Activity	Mapping Data Flow
Purpose	Moves data between sources	Transforms data before loading
Code-Free?	Yes	Yes
Performance	Faster for direct data movement	Uses Spark for transformations
Use Case	ETL/ELT between data stores	Cleansing, joins, aggregations

✓ **Use Copy Data** for **basic ETL**
✓ **Use Data Flow** for **complex transformations**

🔷 Scenario-Based ADF Questions!

1. How would you design an ADF pipeline for daily sales reporting?

💡 **Answer:**
1️⃣ **Source:** Extract data from **on-prem SQL Server**
2️⃣ **Transformation:** Use **Mapping Data Flow** for data cleansing
3️⃣ **Destination:** Load transformed data into **Azure Synapse**
4️⃣ **Trigger:** Schedule a **daily trigger at midnight**
5️⃣ **Monitoring:** Enable **Azure Monitor alerts** for failures

2. How do you implement incremental data load in ADF?

💡 **Answer:**
✅ **Use Watermark Columns** – Track last modified timestamps
✅ **Query Only New/Changed Records** – Use WHERE LastUpdated > @LastRunTime
✅ **Store Last Run Time** – Save in **Azure SQL Table or Blob Metadata**

📌 **Example Query for SQL Incremental Load:**

```
SELECT * FROM SalesData
WHERE LastUpdated > (SELECT MAX(LastRunTime) FROM ADF_Metadata)
```

🔹 Summary: Key ADF Topics to Prepare

Topic	Key Areas to Focus On
ADF Basics	Pipelines, Activities, Datasets, Triggers
Integration Runtimes	Azure IR, Self-Hosted IR, SSIS IR
Data Transformation	Mapping Data Flow, Databricks, Stored Procedures
Security & Monitoring	Managed Identity, Key Vault, Azure Monitor
Error Handling	Retry policies, Logging, Alerts, Custom Handling
Performance Optimization	Partitioning, Caching, Parallel Execution
Real-World Scenarios	Incremental Load, Data Lake Ingestion, Hybrid Data Movement

1. ETL Pipeline: Ingesting and Transforming Sales Data

🔹 **Scenario:**
A retail company wants to **ingest sales data from on-prem SQL Server**, **clean and transform it**, and **store it in Azure Synapse Analytics for reporting**.

🔹 **Solution Approach:**
✅ **Extract:** Copy sales data from SQL Server to Azure Data Lake (ADLS Gen2).
✅ **Transform:** Use **Mapping Data Flow** or **Databricks** for cleansing.
✅ **Load:** Store the transformed data into **Azure Synapse** for Power BI reporting.

🔹 **Step-by-Step Implementation:**
1️⃣ **Create a Linked Service** to connect to **SQL Server** (on-prem).
2️⃣ **Use Self-Hosted IR** to securely move data from on-prem to cloud.
3️⃣ **Copy Data Activity** → Move data to **Azure Data Lake Storage (ADLS Gen2)**.
4️⃣ **Mapping Data Flow** → Clean missing values, format dates, and filter records.
5️⃣ **Load Data into Azure Synapse Analytics** for BI reporting.
6️⃣ **Schedule Pipeline Execution** using **Schedule Trigger** (runs daily at midnight).

📌 **Example Query for Incremental Load:**

```
SELECT * FROM SalesData
WHERE LastUpdated > (SELECT MAX(LastRunTime) FROM ADF_Metadata)
```

✓ **Outcome:** Automated ETL pipeline keeps data updated in **Azure Synapse** for Power BI reports.

2. Data Migration: Moving Data from On-Prem SQL Server to Azure

🔹 **Scenario:**
A financial company wants to **migrate historical data from on-prem SQL Server to Azure SQL Database**.

🔹 **Solution Approach:**
✅ **Extract:** Read data from **on-prem SQL Server**.
✅ **Transfer:** Use **Self-Hosted IR** to securely move data to Azure.
✅ **Load:** Store data in **Azure SQL Database** with **incremental updates**.

🔹 **Step-by-Step Implementation:**
1️⃣ **Install Self-Hosted Integration Runtime (SHIR)** on an on-prem machine.
2️⃣ **Create a Linked Service** to connect **SQL Server and Azure SQL Database**.
3️⃣ **Use Copy Data Activity** to transfer data.
4️⃣ **Enable Incremental Load** using **Watermark Columns**.
5️⃣ **Monitor & Log Pipeline Runs** using **Azure Monitor**.

📌 **Example Query for Incremental Load (Watermarking Approach):**

```
SELECT * FROM Transactions
WHERE LastUpdated > (SELECT MAX(LastProcessedDate) FROM MigrationLog)
```

✓ **Outcome:** On-prem data is seamlessly migrated and updated in **Azure SQL**.

3. Real-Time Data Processing from IoT Devices

🔹 **Scenario:**
A manufacturing company wants to **process IoT sensor data in real-time** and **store it in Azure Data Lake for analytics**.

🔹 **Solution Approach:**

✅ **Ingest Data from IoT Hub** using **Event-Based Triggers** in ADF.

✅ **Transform Data in Databricks** – Aggregate, filter, and cleanse data.

✅ **Store Processed Data in Delta Lake** for analytics.

🔹 **Step-by-Step Implementation:**

1️⃣ **Use Event-Based Trigger** to detect new IoT data arrival in **ADLS**.

2️⃣ **Copy Raw IoT Data** to **Azure Databricks** for processing.

3️⃣ **Use Databricks Notebooks** to filter anomalies, aggregate sensor readings.

4️⃣ **Store Data in Delta Lake** (Optimized for analytics).

5️⃣ **Use Power BI for Real-Time Dashboards**.

📌 **Example PySpark Code for IoT Data Processing in Databricks:**

```
from pyspark.sql.functions import avg, col


df = spark.read.json("dbfs:/mnt/iot/raw/")

df_cleaned = df.filter(col("temperature").isNotNull()) \

        .groupBy("device_id") \

        .agg(avg("temperature").alias("avg_temperature"))

df_cleaned.write.format("delta").save("dbfs:/mnt/iot/processed/")
```

✓ **Outcome:** Real-time IoT data is processed and visualized in **Power BI dashboards**.

4. Automating Data Ingestion from REST APIs

🔹 **Scenario:**

A healthcare company needs to **fetch patient records from a third-party API**, process them, and store them in **Azure SQL Database**.

🔹 **Solution Approach:**

✅ **Extract Data from API** using **Web Activity in ADF**.

✅ **Transform Data** in **Mapping Data Flow** (clean, remove duplicates).

✅ **Load Data into Azure SQL Database** for reporting.

🔹 **Step-by-Step Implementation:**

1️⃣ **Create a Web Activity** in ADF to call REST API (GET request).

2️⃣ **Store JSON response in ADLS** for staging.
3️⃣ **Use Mapping Data Flow** to parse and clean API data.
4️⃣ **Use Copy Data Activity** to store data in **Azure SQL**.
5️⃣ **Schedule Pipeline Execution** using **Triggers**.

📌 **Example API Request in ADF Web Activity:**

```
{
  "url": "https://api.example.com/patients",
  "method": "GET",
  "headers": {
    "Authorization": "Bearer XYZ123"
  }
}
```

✓ **Outcome:** API data is automatically fetched and stored in **Azure SQL** for further analysis.

5. Processing and Storing Large CSV Files in ADLS

🔹 **Scenario:**
A logistics company receives **large CSV files daily** containing shipment data. The goal is to **store them in Azure Data Lake and optimize for fast querying**.

🔹 **Solution Approach:**
✅ **Ingest CSV Files** using **Event-Based Triggers** in ADF.
✅ **Convert CSV to Parquet Format** for better performance.
✅ **Store in Azure Data Lake & Query with Synapse**.

🔹 **Step-by-Step Implementation:**
1️⃣ **Use Event-Based Trigger** to detect new CSV files in ADLS.
2️⃣ **Copy Data Activity** to move raw CSV files to **staging folder**.
3️⃣ **Use Mapping Data Flow** to convert CSV to **Parquet format**.
4️⃣ **Store Processed Data in Azure Data Lake (ADLS Gen2)**.
5️⃣ **Query Data Using Azure Synapse Serverless SQL**.

📌 **Example Query to Read Parquet Data in Synapse:**

```
SELECT * FROM OPENROWSET(
   BULK 'https://datalake.blob.core.windows.net/processed/shipments.parquet',
   FORMAT='PARQUET'
) AS Shipments
```

✓ **Outcome: Optimized Parquet files allow faster queries and reduced storage costs.**

Gopi Rayavarapu