The **Medallion Architecture** is a data design pattern used in data engineering, particularly in **lakehouse environments**. It organizes data into three layers—Bronze, Silver, and Gold—each representing a different stage of data refinement and quality. Here's how it works:

1. **Bronze Layer (Raw Data)**:
   * This is where all the raw data from external sources is ingested.
   * The data is stored "as-is," without any processing or transformation.
   * It serves as a historical archive and provides data lineage and auditability.
   * Example: Logs from applications or streaming events from Kafka.
2. **Silver Layer (Cleansed Data)**: (transformation - using databricks )
   * Data from the Bronze layer is cleaned, validated, and conformed in this layer.
   * It provides an enterprise view of key business entities, such as master customers or transactions.
   * Minimal transformations are applied to make the data usable for analytics and machine learning.
   * Example: Removing duplicates, handling null values, and merging datasets.
3. **Gold Layer (Enriched Data)**(
   * This layer contains aggregated and enriched data, ready for consumption by business intelligence tools and machine learning models.
   * Complex transformations and business rules are applied here.
   * Example: Dimensional modeling, creating dashboards, or predictive analytics.

**Difference between blobstorage & Data lakes**

 A**zure Blob Storage:**

* Blob storage is a general-purpose object storage solution in Azure.
* It's used to store large amounts of unstructured data like text or binary data.
* Typical use cases include backups, media files, logs, and simple archival.

 **Azure Data Lake Storage (ADLS)**

**(\*\*\* in azure we need to slect hierarchy for this to use other wise it will create an blob storage \*\*\*)**

* Data Lake Storage builds on Azure Blob storage but provides specialized capabilities for **big data analytics**.
* It offers hierarchical file system capabilities (like directories and folders), which make it ideal for structured, semi-structured, and unstructured data.
* It's better suited for scenarios involving processing massive datasets using distributed systems like Hadoop or Spark.

IMP - in architecture diagram right we have use data factory and data lakes so why we cant directly read the data from the

Ans- Azure Data Factory shines because it can **read multiple types of data from various sources** in parallel, handle the complexities of different formats, and process them efficiently—whether it's databases, APIs, files, or streams.

As for **data lakes**, you're absolutely right again—directly reading or managing data in a data lake can be **complex** without a tool to orchestrate or optimize the process. ADF simplifies this by automating, scaling, and organizing the data ingestion workflows, making your life a lot easier.

 **Linked Service**: Think of it as a connection string. It defines the connection information needed for Data Factory to connect to external resources, such as databases, storage accounts, APIs, or other services. For example, if you're pulling data from an Azure SQL Database, the Linked Service would store the credentials and connection details for that database.

 **Data Source**: This refers to the actual location or system where your data resides. It could be a database, a flat file, a cloud storage service, or even live data streams. Data Sources are what you interact with through Linked Services to extract, transform, and load (ETL) data.

**Manual Pipelines:**

* Multiple datasets for each file or source.
* Static file paths and repetitive effort.
* Not scalable—manual updates needed for new files/sources.

**Dynamic Pipelines:**

* Single dataset with **parameters** to handle multiple files/sources.
* Uses **Foreach loops** to process multiple inputs dynamically.
* Scalable, efficient, and cost-effective.

**Key Advantages of Dynamic Pipelines:**

1. Flexibility with parameterized datasets.
2. Automation reduces manual effort.
3. Reusable pipeline templates.
4. Handles any data format dynamically.