**Madiha Aimon Tappal**

[madihaaimon@gmail.com](mailto:madihaaimon@gmail.com)

**Data Engineering Batch – 1**

**Day – 19 Assignment**

**Azure Databricks**

**What is a database?**

A database is a collection of data or information. Databases are typically accessed electronically and are used to support Online Transaction Processing (OLTP). Database Management Systems (DBMS) store data in the database and enable users and applications to interact with the data. The term “database” is commonly used to reference both the database itself as well as the DBMS.

A database is a structured collection of data that is organized and stored in a way that facilitates efficient retrieval and management of information. It is designed to provide a systematic and efficient method for data storage, retrieval, and management. Databases are widely used in various applications and industries to store, organize, and manipulate large volumes of data.

Key characteristics of a database include:

1. **Structured Data:** Databases store data in a structured format, typically using tables, rows, and columns. This structured approach allows for efficient organization and retrieval of information.
2. **Data Integrity:** Databases enforce data integrity by defining rules and constraints on the stored data. This ensures that the data remains accurate and consistent over time.
3. **Data Independence:** Databases allow for the separation of data from the applications that use it. This means that changes to the structure or organization of the data do not impact the applications that access it, providing a level of abstraction.
4. **Query Language:** Databases use a query language (such as SQL - Structured Query Language) to interact with and retrieve data. This allows users to perform complex searches and manipulations on the data.
5. **Concurrency Control:** Databases manage concurrent access to data by multiple users or applications, ensuring that transactions are executed in a controlled and consistent manner to maintain data integrity.
6. **Transaction Support:** Databases support transactions, which are sequences of one or more operations that are executed as a single unit. Transactions ensure that either all the operations are completed successfully or none of them are, maintaining consistency.
7. **Scalability:** Databases are designed to scale, allowing them to handle increasing amounts of data and user requests. Scalability can be achieved through techniques like partitioning, sharding, and replication.
8. **Security:** Databases provide mechanisms to control access to the data, ensuring that only authorized users or applications can retrieve or modify specific information. Security features also include encryption, authentication, and auditing.
9. **Backup and Recovery:** Databases include features for creating backups of the data to prevent loss in case of system failures or disasters. They also provide mechanisms for recovering data to a consistent state after a failure.
10. **Data Relationships:** Databases allow the establishment of relationships between different sets of data, enabling the representation of complex structures and connections within the information.

These characteristics collectively contribute to the effectiveness and reliability of databases in managing and manipulating data in a variety of applications.

**Database characteristics: -**

A variety of database types have emerged over the last several decades. All databases store information, but each database will have its own characteristics. Relational databases store data in tables with fixed rows and columns. Non-relational databases (also known as NoSQL databases) store data in a variety of models including JSON (JavaScript Object Notation), BSON (Binary JSON), key-value pairs, tables with rows and dynamic columns, and nodes and edges. Databases store structured and/or semi-structured data, depending on the type.

You may also find database characteristics like:

* Security features to ensure the data can only be accessed by authorized users.
* ACID (Atomicity, Consistency, Isolation, Durability) transactions to ensure data integrity.
* Query languages and APIs to easily interact with the data in the database.
* Indexes to optimize query performance.
* Full-text search.
* Optimizations for mobile devices.
* Flexible deployment topologies to isolate workloads (e.g., analytics workloads) to a specific set of resources.
* On-premises, private cloud, public cloud, hybrid cloud, and/or multi-cloud hosting options.

### Database examples: -

A myriad of databases exist. Examples include:

* Relational databases: Oracle, MySQL, Microsoft SQL Server, and PostgreSQL
* Document databases: MongoDB and CouchDB
* Key-value databases: Redis and DynamoDB
* Wide-column stores: Cassandra and HBase
* Graph databases: Neo4j and Amazon Neptune

## OLAP + data warehouses and data lakes

Both data warehouses and data lakes are meant to support Online Analytical Processing (OLAP). OLAP systems are typically used to collect data from a variety of sources. The data is then used to power a range of analytical use cases ranging from business intelligence and reporting (e.g., quarterly sales reports by store) to forecasting (e.g., predicting home sales for the next six months based on historical trends).

## What is a data warehouse?

A data warehouse is a system that stores highly structured information from various sources. Data warehouses typically store current and historical data from one or more systems. The goal of using a data warehouse is to combine disparate data sources in order to analyze the data, look for insights, and create business intelligence (BI) in the form of reports and dashboards.

You might be wondering, "Is a data warehouse a database?" Yes, a data warehouse is a giant database that is optimized for analytics.

### Data warehouse characteristics: -

Data warehouses store large amounts of current and historical data from various sources. They contain a range of data, from raw ingested data to highly curated, cleansed, filtered, and aggregated data.

Extract, transform, load (ETL) processes move data from its original source to the data warehouse. The ETL processes move data on a regular schedule (for example, hourly or daily), so data in the data warehouse may not reflect the most up-to-date state of the systems.

Data warehouses typically have a pre-defined and fixed relational schema. Therefore, they work well with structured data. Some data warehouses also support semi-structured data.

Once the data is in the warehouse, business analysts can connect data warehouses with BI tools. These tools allow business analysts and data scientists to explore the data, look for insights, and generate reports for business stakeholders.

### Why use a data warehouse?

Data warehouses are a good option when you need to store large amounts of historical data and/or perform in-depth analysis of your data to generate business intelligence. Due to their highly structured nature, analyzing the data in data warehouses is relatively straightforward and can be performed by business analysts and data scientists.

Note that data warehouses are not intended to satisfy the transaction and concurrency needs of an application. If an organization determines they will benefit from a data warehouse, they will need a separate database or databases to power their daily operations.

### Data warehouse examples: -

Examples of data warehouses include:

* Amazon Redshift.
* Google BigQuery.
* IBM Db2 Warehouse.
* Microsoft Azure Synapse.
* Oracle Autonomous Data Warehouse.
* Snowflake.
* Teradata Vantage.

## What is a data lake?

A data lake is a repository of data from disparate sources that is stored in its original, raw format. Like data warehouses, data lakes store large amounts of current and historical data. What sets data lakes apart is their ability to store data in a variety of formats including JSON, BSON, CSV, TSV, Avro, ORC, and Parquet.

Typically, the primary purpose of a data lake is to analyze the data to gain insights. However, organizations sometimes use data lakes simply for their cheap storage with the idea that the data may be used for analytics in the future.

### Is a data lake a database?

You might be wondering, "Is a data lake a database?" A data lake is a repository for data stored in a variety of ways including databases. With modern tools and technologies, a data lake can also form the storage layer of a database. Tools like Starburst, Presto, Dremio, and Atlas Data Lake can give a database-like view into the data stored in your data lake. In many cases, these tools can power the same analytical workloads as a data warehouse.

### Data lake characteristics: -

Data lakes store large amounts of structured, semi-structured, and unstructured data. They can contain everything from relational data to JSON documents to PDFs to audio files.

Data does not need to be transformed in order to be added to the data lake, which means data can be added (or “ingested”) incredibly efficiently without upfront planning.

The primary users of a data lake can vary based on the structure of the data. Business analysts will be able to gain insights when the data is more structured. When the data is more unstructured, data analysis will likely require the expertise of developers, data scientists, or data engineers.

The flexible nature of data lakes enables business analysts and data scientists to look for unexpected patterns and insights. The raw nature of the data combined with its volume allows users to solve problems they may not have been aware of when they initially configured the data lake.

Data in data lakes can be processed with a variety of OLAP systems and visualized with BI tools.

### Why use a data lake?

Data lakes are a cost-effective way to store huge amounts of data. Use a data lake when you want to gain insights into your current and historical data in its raw form without having to transform and move it. Data lakes also support machine learning and predictive analytics.

Like data warehouses, data lakes are not intended to satisfy the transaction and concurrency needs of an application.

### Data lake examples

Data lakes can provide storage and compute capabilities, either independently or together.

The following are examples of technology that provide flexible and scalable storage for building data lakes:

* AWS S3
* Azure Data Lake Storage Gen2(curriculum included)
* Google Cloud Storage

Other technologies enable organizing and querying data in data lakes, including:

* MongoDB Atlas Data Lake.
* AWS Athena.
* Presto.
* Starburst.
* Databricks SQL Analytics.

**What are the key differences between a database, data warehouse, and data lake?**

| **Characteristic** | **Database** | **Data Warehouse** | **Data Lake** |
| --- | --- | --- | --- |
| **Purpose** | Transactional data storage | Analytical querying and reporting | Store raw, unstructured data |
| **Data Type** | Structured data | Structured and semi-structured | Structured, semi-structured, and unstructured data |
| **Schema** | Schema-on-write | Schema-on-write | Schema-on-read |
| **Storage Optimization** | Optimized for read and write | Optimized for read performance | Cost-effective storage with flexibility in data types |
| **Query Performance** | High-speed transaction processing | Optimized for complex queries | Query performance depends on processing frameworks |
| **Data Processing** | Relational database management system (RDBMS) | ETL (Extract, Transform, Load) processes for data transformation | Supports various processing frameworks (e.g., Spark, Hadoop) |
| **Data History** | Keeps current data | Retains historical data | Stores raw data with versioning capabilities |
| **Latency** | Low latency for real-time transactions | Variable latency depending on data complexity | Variable latency, suitable for batch and real-time processing |
| **Use Cases** | OLTP (Online Transaction Processing) | OLAP (Online Analytical Processing) | Big data analytics, machine learning, exploratory analysis |
| **Examples** | MySQL, PostgreSQL, Oracle | Amazon Redshift, Snowflake, Google BigQuery | Hadoop Distributed File System (HDFS), Amazon S3 |

| **Pros** | **Database** | **Data Warehouse** | **Data Lake** |
| --- | --- | --- | --- |
| **Pros** | Fast transaction processing | High-performance analytics | Flexibility in handling diverse data types |
|  | Structured data integrity | Historical data analysis | Scalability for massive data storage |
|  | ACID compliance for transactions | Data consolidation from various sources | Supports both batch and real-time processing |
|  | Ease of use and familiarity with SQL | Optimized for reporting | Cost-effectiveness for storing large volumes of raw data |
|  | Data indexing for quick retrieval | Scalability for large datasets | Ability to store raw data without predefined structure |
|  | Security features for access control | Centralized data repository | Supports schema evolution over time |
|  |  |  | Integration with various processing frameworks |
|  |  |  | Parallel processing for distributed computing |

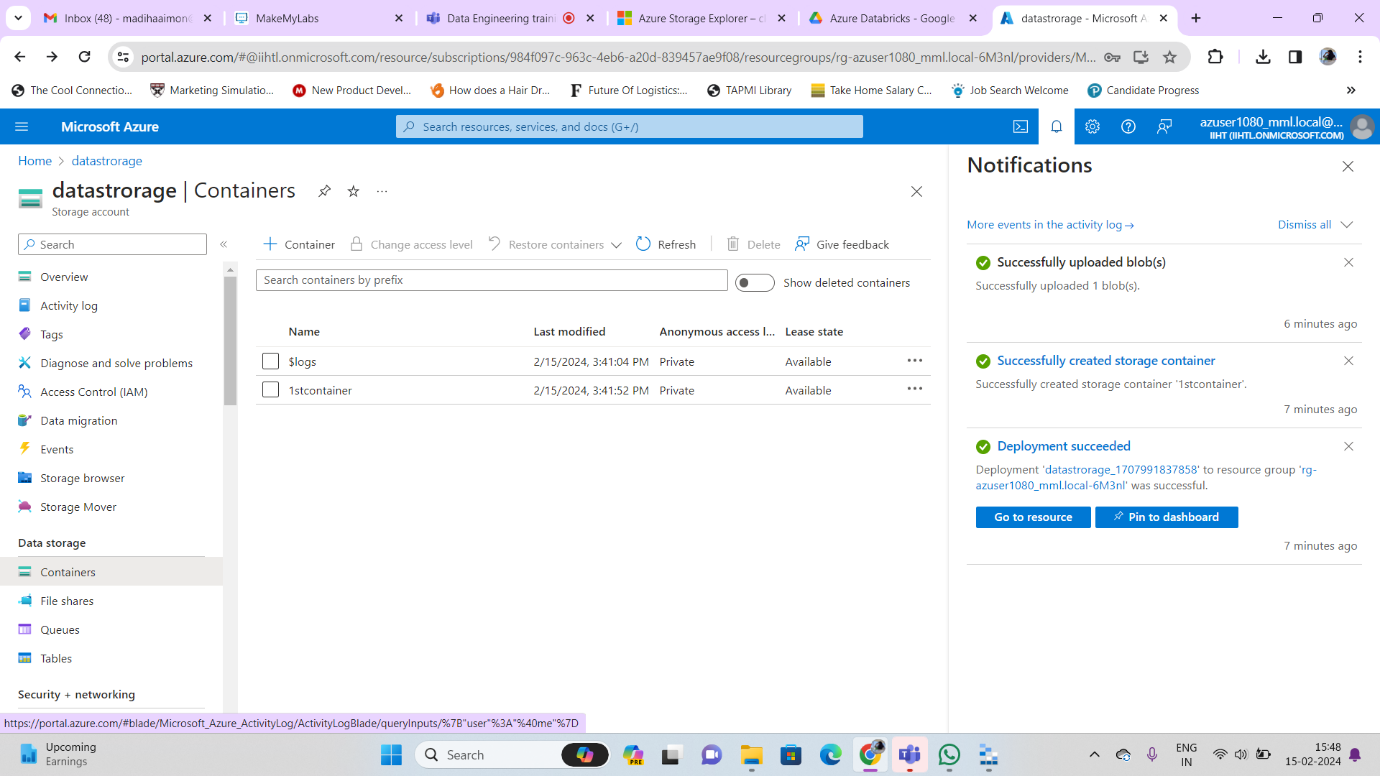
| **Cons** | **Database** | **Data Warehouse** | **Data Lake** |
| --- | --- | --- | --- |
| **Cons** | Limited support for complex analytics | Cost of storage and processing | Complexity in managing schema-on-read |
|  | Scaling challenges for large datasets | Initial setup and maintenance costs | Potential for data redundancy and inconsistency |
|  | May not handle large-scale analytics | Complexity in data transformation | Requires specialized skills for data processing frameworks |
|  | Limited flexibility for handling diverse data types | May not support real-time analytics | Query performance may vary based on processing frameworks |
|  | Limited scalability for massive data storage | Latency in accessing and retrieving data | Potential for increased storage costs without proper governance |
|  |  | Limited support for raw, unstructured data | Governance challenges without proper metadata management |
|  |  |  | Requires effective data governance for security and compliance |

Hands On Practice

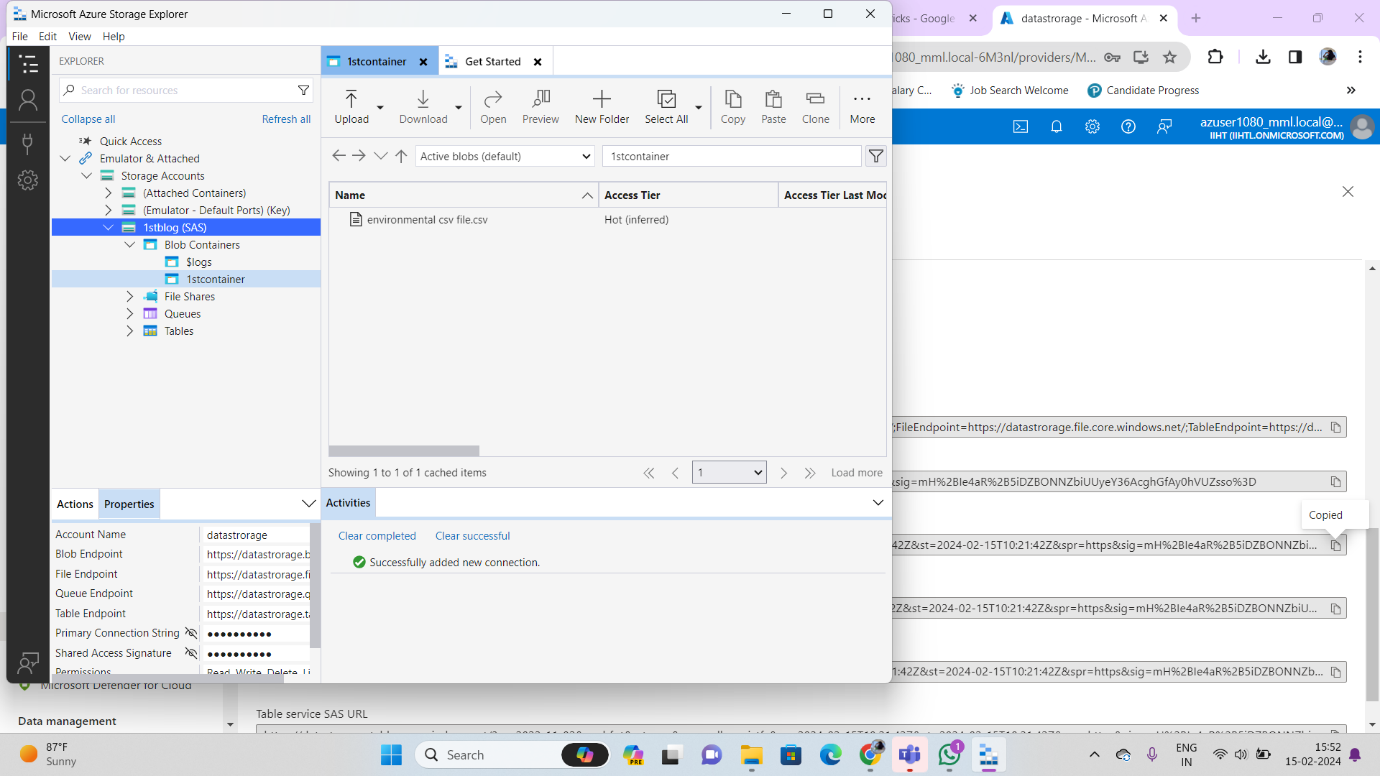
Creating storage class in Azure : -

1.Container creating

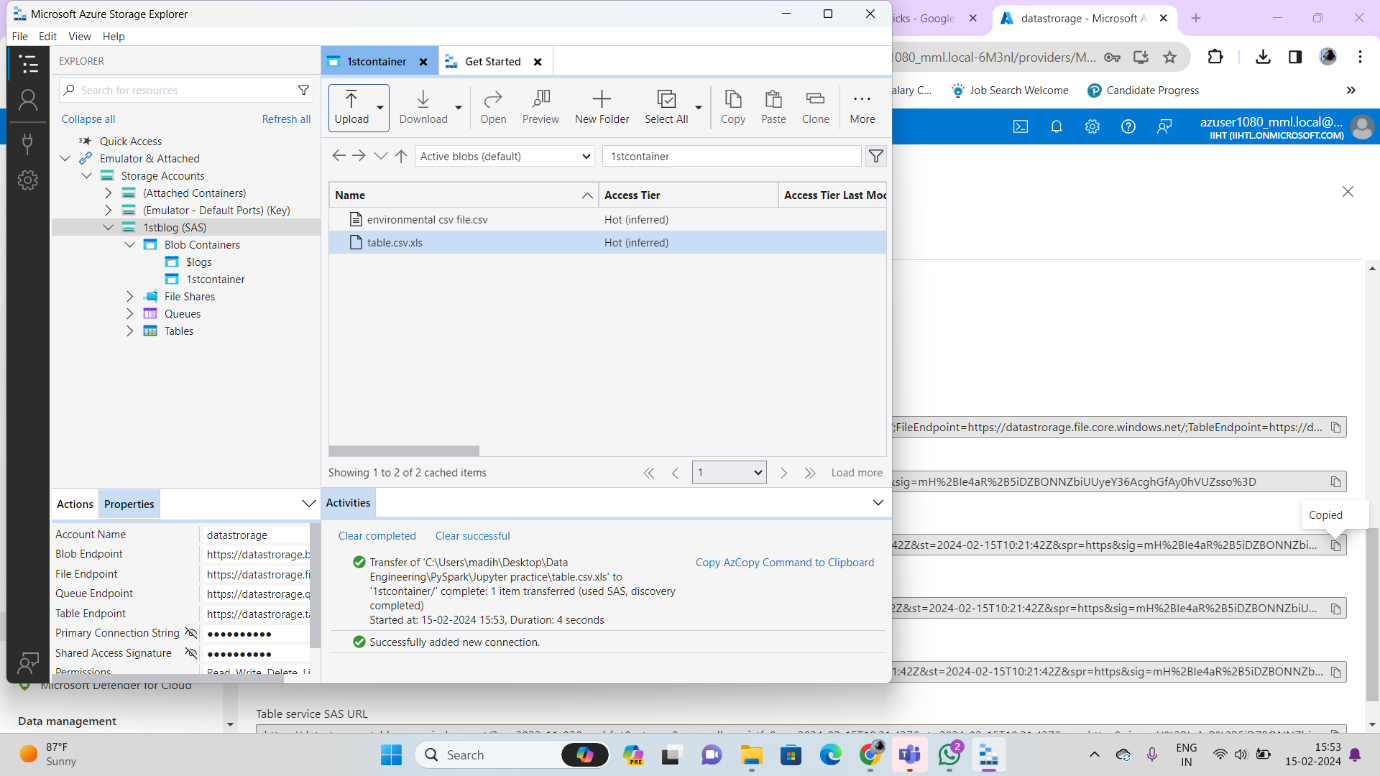
2. Adding files to container



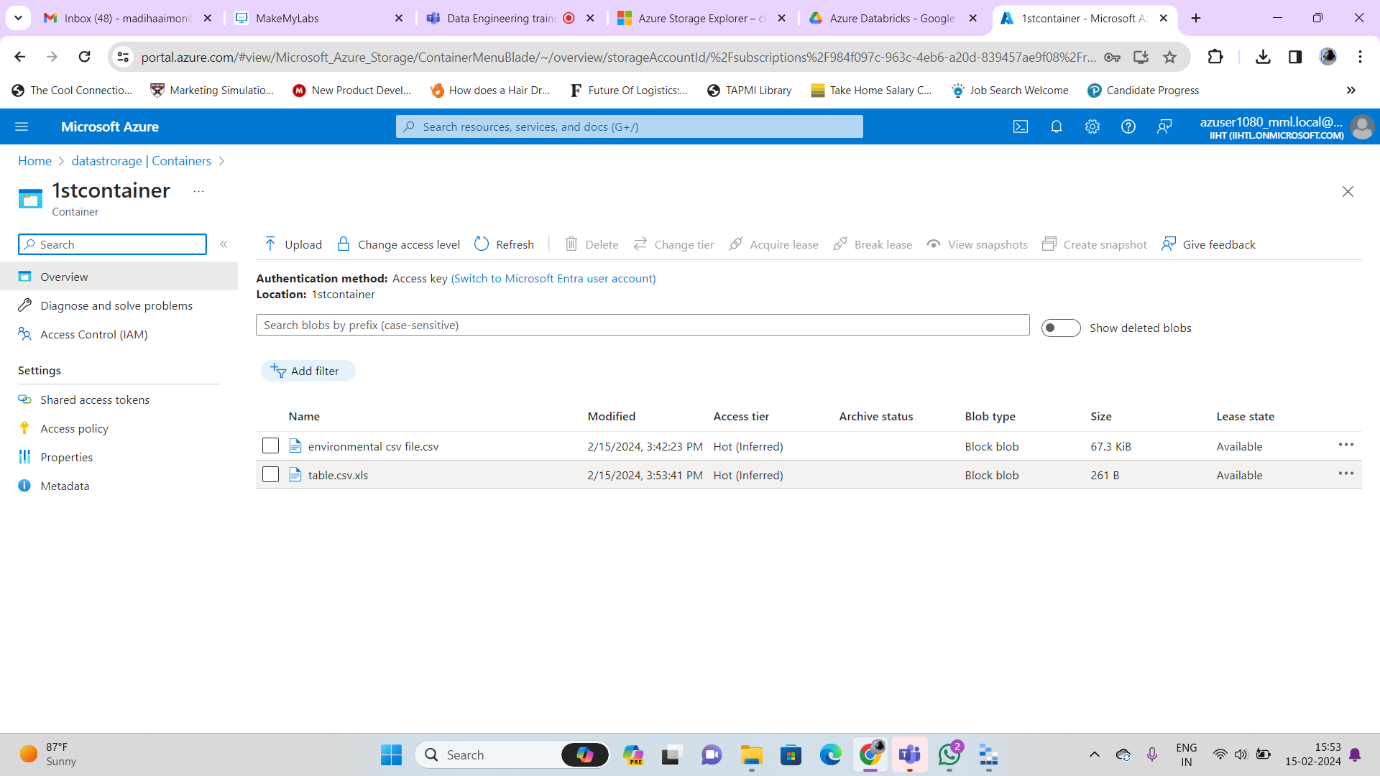
Connecting data storage to local:



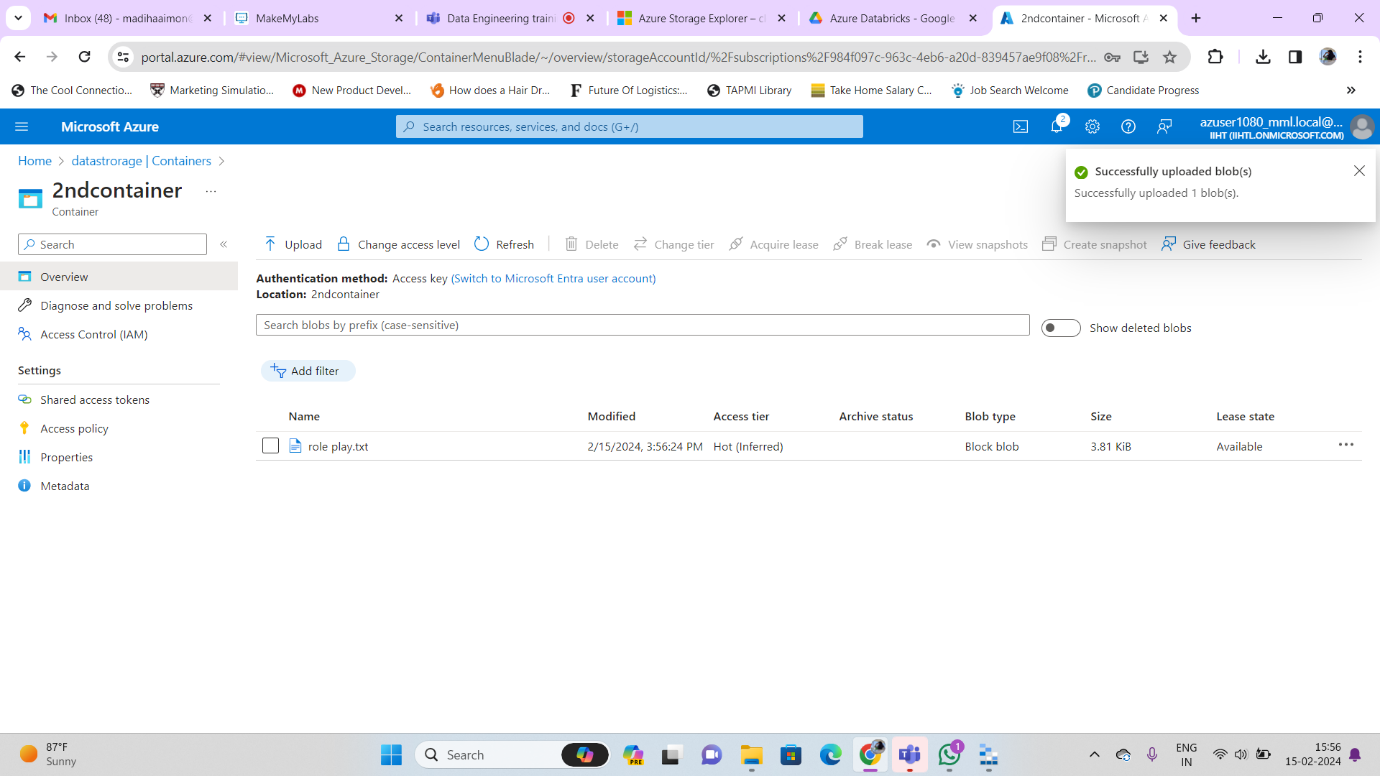
files uploaded in portal will reflect on your local



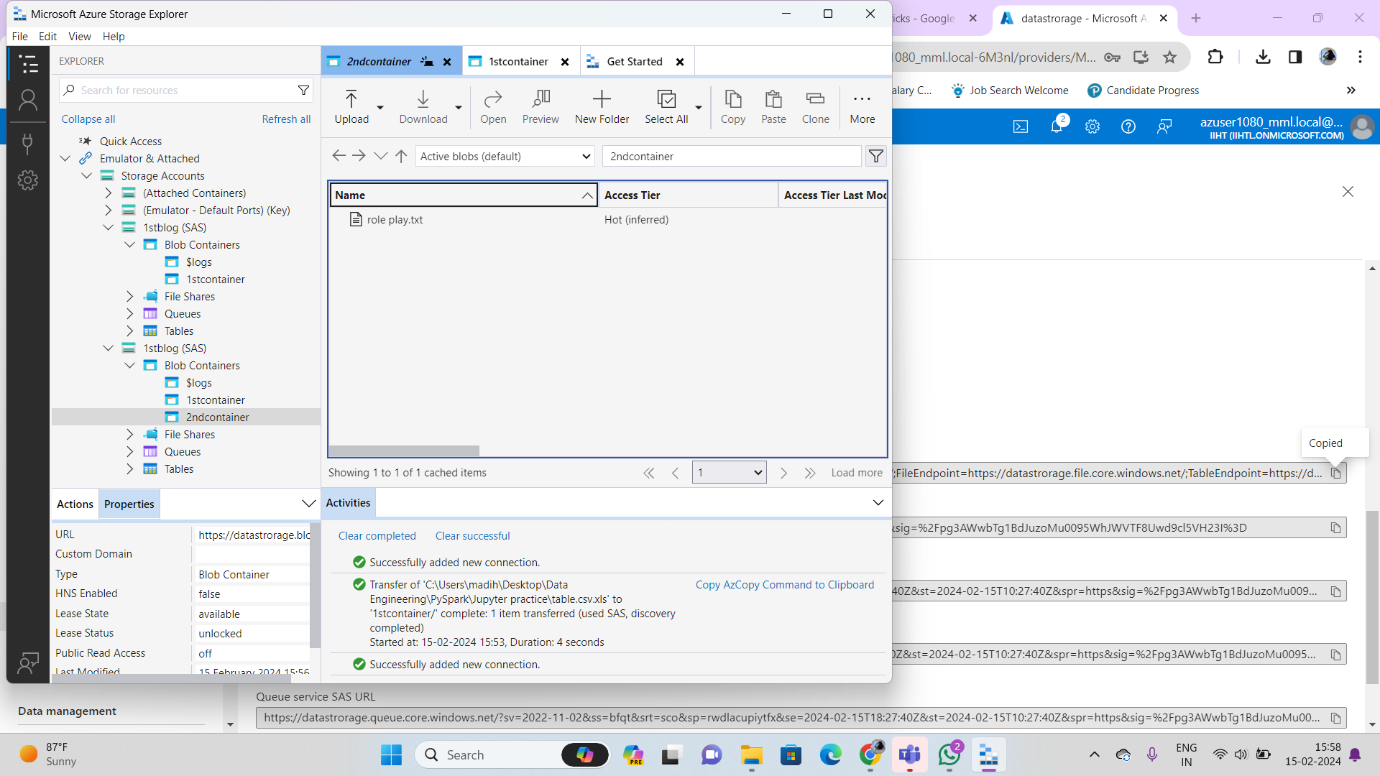
Check in portal



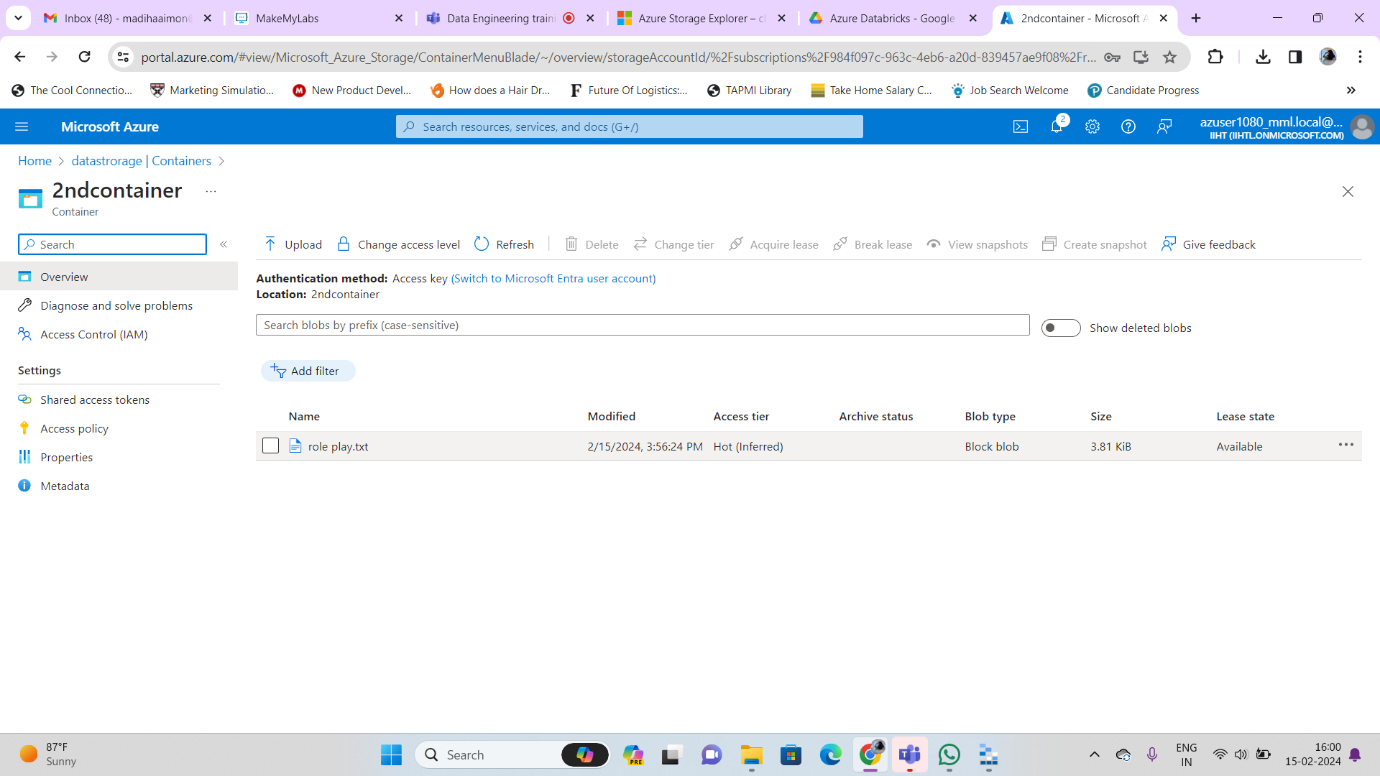
Create 2nd container in same storage class and add files



Same container and files will reflets in local



In potal



**Notes written:**

