Non-relational data services (NoSQL databases) in **Microsoft Azure** provide highly scalable and flexible storage solutions for unstructured and semi-structured data. Azure offers multiple NoSQL database services, including **Azure Cosmos DB, Azure Table Storage, and Azure Blob Storage**, each tailored for specific workloads.

**Key Non-Relational Data Services in Azure**

| **Service** | **Description** |
| --- | --- |
| **Azure Cosmos DB** | Fully managed NoSQL database supporting multiple APIs (SQL, MongoDB, Cassandra, Gremlin, Table). |
| **Azure Table Storage** | Key-Value store for structured NoSQL data, optimized for scalability. |
| **Azure Blob Storage** | Object storage for large-scale unstructured data (images, videos, logs). |

**Configuring Azure Cosmos DB**

**Step 1: Create an Azure Cosmos DB Account**

1. **Log in to** [**Azure Portal**](https://portal.azure.com).
2. Search for **"Azure Cosmos DB"** and select **Create**.
3. Choose the API type based on your use case:
   * **Core (SQL API)** – Document-based queries.
   * **MongoDB API** – MongoDB-compatible NoSQL.
   * **Cassandra API** – Column-family store.
   * **Gremlin API** – Graph database.
   * **Table API** – Key-Value storage.
4. Configure **Subscription, Resource Group, Account Name, Region, and Capacity Mode (Provisioned or Serverless)**.
5. Click **Review + Create** → **Create**.

**Step 2: Create a Database and Container**

1. Navigate to the **Cosmos DB account** you created.
2. Go to **Data Explorer** → Click **New Database**.
3. Define **Database ID** and select **Throughput (e.g., 400 RU/s)**.
4. Click **OK**.
5. Create a **New Container** inside the database:
   * **Container ID**: Name of the collection/table.
   * **Partition Key**: e.g., /customerId.
   * Click **OK** to create the container.

**Step 3: Connect to Cosmos DB**

**Using Connection String**

1. Go to **Azure Cosmos DB** → **Keys**.
2. Copy **Primary Connection String**.

**Example Connection with Python (MongoDB API)**

python

Sao chépChỉnh sửa

from pymongo import MongoClient

client = MongoClient("your-cosmos-db-connection-string")

db = client["your-database-name"]

collection = db["your-collection-name"]

# Insert data

collection.insert\_one({"customerId": "123", "name": "Alice", "order": "Laptop"})

**Configuring Azure Table Storage**

**Step 1: Create an Azure Storage Account**

1. Go to **Azure Portal** → Search **"Storage Account"** → Click **Create**.
2. Configure **Subscription, Resource Group, Storage Account Name, Region, and Performance Tier**.
3. Choose **Redundancy Level (LRS, GRS, ZRS, RA-GRS)**.
4. Click **Review + Create** → **Create**.

**Step 2: Create a Table Storage**

1. Navigate to the **Storage Account** → Click **Tables**.
2. Click **+ Table**, enter a name, and create it.

**Step 3: Connect and Insert Data in Table Storage**

**Using Connection String**

1. Go to **Storage Account** → **Access Keys** → Copy **Connection String**.

**Example Connection with Python (Azure Table Storage SDK)**

from azure.data.tables import TableServiceClient, TableEntity

connection\_string = "your-azure-storage-connection-string"

table\_service = TableServiceClient.from\_connection\_string(conn\_str=connection\_string)

table\_client = table\_service.get\_table\_client("your-table-name")

# Insert data

entity = TableEntity()

entity["PartitionKey"] = "Customers"

entity["RowKey"] = "123"

entity["Name"] = "John Doe"

table\_client.create\_entity(entity)

**Configuring Azure Blob Storage**

**Step 1: Create a Blob Storage Container**

1. In **Storage Account**, go to **Containers**.
2. Click **+ Container**, enter a name, and set **access level (Private/Public)**.
3. Click **Create**.

**Step 2: Upload Data to Blob Storage**

**Using Python SDK to Upload a File**

from azure.storage.blob import BlobServiceClient

connection\_string = "your-azure-storage-connection-string"

blob\_service\_client = BlobServiceClient.from\_connection\_string(connection\_string)

container\_client = blob\_service\_client.get\_container\_client("your-container-name")

# Upload a file

with open("data.txt", "rb") as data:

blob\_client = container\_client.get\_blob\_client("data.txt")

blob\_client.upload\_blob(data)