Provisioning a **non-relational data service** in Azure involves setting up a scalable NoSQL database or storage solution to handle **unstructured and semi-structured data**. Azure provides multiple **NoSQL services**, including **Azure Cosmos DB, Azure Table Storage, and Azure Blob Storage**, each designed for different use cases.

**Steps for Provisioning Non-Relational Data Services**

Provisioning typically includes:

1. **Creating the service** (e.g., Cosmos DB, Table Storage, Blob Storage).
2. **Configuring storage and access policies**.
3. **Connecting applications to the service** using SDKs or APIs.
4. **Managing scaling and performance settings**.

**Provisioning Azure Cosmos DB**

Azure Cosmos DB is a **multi-model NoSQL database** designed for **global distribution and high scalability**.

**🔹 Step 1: Create a Cosmos DB Account**

1. **Go to** [**Azure Portal**](https://portal.azure.com) → Search for **"Azure Cosmos DB"**.
2. Click **Create** → Choose an API:
   * **Core (SQL API)** → Document-based queries.
   * **MongoDB API** → MongoDB-compatible database.
   * **Cassandra API** → Column-family store.
   * **Gremlin API** → Graph database.
   * **Table API** → Key-value storage.
3. Configure **Subscription, Resource Group, Account Name, and Region**.
4. Select **Throughput Mode**:
   * **Provisioned Throughput (400 RU/s or more)** → Best for consistent performance.
   * **Serverless** → Best for intermittent workloads.
5. Click **Review + Create** → **Create**.

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

1. Open the **Cosmos DB account** in Azure.
2. Navigate to **Data Explorer** → Click **New Database**.
3. Define:
   * **Database ID** → Name of the database.
   * **Throughput** → e.g., 400 RU/s (Request Units per second).
4. Click **OK**.
5. Inside the **database**, click **New Container**:
   * **Container ID** → Name of the collection/table.
   * **Partition Key** → e.g., /customerId.
6. Click **OK** to create the container.

**Step 3: Connect an Application to Cosmos DB**

**Using Connection String**

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

**Example Python Connection (MongoDB API)**

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"})

**Provisioning Azure Table Storage**

Azure Table Storage is a **high-performance key-value store** optimized for structured NoSQL data.

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

1. Go to **Azure Portal** → Search for **Storage Account** → Click **Create**.
2. Configure:
   * **Subscription & Resource Group**.
   * **Storage Account Name**.
   * **Region**.
   * **Performance Tier**: **Standard** (cost-effective) or **Premium** (low latency).
   * **Redundancy**: Choose **LRS, GRS, ZRS, or RA-GRS**.
3. Click **Review + Create** → **Create**.

**Step 2: Create a Table Storage**

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

**Step 3: Connect an Application to Table Storage**

**Using Connection String**

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

**Example Python Connection**

python

Sao chépChỉnh sửa

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)

**Provisioning Azure Blob Storage**

Azure Blob Storage is **optimized for unstructured data** (e.g., images, videos, backups, logs).

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

1. Navigate to **Storage Account** → Click **Containers**.
2. Click **+ Container**, enter a **Container Name**.
3. Set **Access Level** (Private, Blob, Container).
4. Click **Create**.

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

**Example Python Code 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)

**Summary: Which Service to Use?**

| **Service** | **Best For** | **Examples** |
| --- | --- | --- |
| **Azure Cosmos DB** | NoSQL apps with high scalability & low latency | IoT, E-commerce, AI apps |
| **Azure Table Storage** | Simple key-value storage | Logging, metadata, session data |
| **Azure Blob Storage** | Large unstructured data | Images, videos, backups |