# **Day 29 - 30 August 2025**

**Document Name:**Day 29 - hmuvvala@ - Hari Gopal Muvvala

## **1. AWS CLI (Command Line Interface)**

* Tool to interact with AWS services using commands instead of AWS Console (GUI).
* Allows automation, scripting, and quicker operations.
* Installed locally on your machine.
* Verifying installation: aws --version
* If not working properly, ensure PATH/environment variables are set correctly.

## **2. Amazon DynamoDB Overview**

* **Type**: Fully managed **NoSQL database** service by AWS.
* **Serverless**: No need to manage servers, scaling, or patching.
* **Highly Available & Durable**: Data is automatically replicated across multiple AZs in a region.
* **Scalable**: Handles large amounts of traffic with high performance.
* **Use Cases**: Web apps, mobile apps, gaming, IoT, real-time analytics.

### **Key Features**

* **Flexible Data Model**:
  + Schema-less tables (no fixed columns like SQL).
  + Items (like rows) made up of attributes (like columns).
  + JSON format is supported.
* **Consistency Models**:
  + Strongly consistent reads (latest data).
  + Eventually consistent reads (faster, might be slightly outdated).
* **Indexes**:
  + **Primary Key** = Partition Key (+ optional Sort Key).
  + **Local Secondary Index (LSI)** – same partition key, different sort key.
  + **Global Secondary Index (GSI)** – different partition key and sort key.
* **Capacity Modes**:
  + **Provisioned**: Pre-define read/write capacity units (RCUs & WCUs).
  + **On-Demand (PAY-PER-REQUEST)**: Automatically scales based on usage.

## **3. DynamoDB vs SQL (RDBMS)**

| **RDBMS (SQL)** | **DynamoDB (NoSQL)** |
| --- | --- |
| Uses **tables, rows, columns** | Uses **tables, items, attributes** |
| Strong schema | Schema-less (flexible attributes per item) |
| Connects using persistent SQL connection | Connects via **HTTP APIs** |
| Complex queries, joins supported | Simple queries, filtering via keys |
| Vertical scaling (increase single server size) | Horizontal scaling (auto-partitioning across servers) |

## **4. Operations in DynamoDB**

* **Create Table** – Define table name, attributes, and keys.
* **Insert/Put Item** – Add a new item (row equivalent).
* **Read/Get Item / Scan / Query** – Fetch data (by key or filtering).
* **Update Item** – Modify attributes of existing item.
* **Delete Item** – Remove data.
* **Delete Table** – Remove entire table.

## **5. CAP Theorem (Applied to DynamoDB)**

* **Consistency** – All nodes see the same data at the same time.
* **Availability** – Every request receives a response (even if not the latest data).
* **Partition Tolerance** – System continues to operate even if network issues occur.
* DynamoDB focuses on **high availability and partition tolerance**, but you can choose strong or eventual consistency.

## **6. NoSQL Workbench (for DynamoDB)**

* A **GUI tool** provided by AWS to visually model, design, and query DynamoDB.
* Easier for beginners compared to CLI.
* Can create tables, insert items, and run queries directly.

## **7. DynamoDB Local & SDK**

* **DynamoDB Local**: Run DynamoDB on your machine for testing without using real AWS resources.
  + Runs via a JAR file (DynamoDBLocal.jar).
  + Useful for development, avoids AWS charges.
* **AWS SDK**: Software Development Kit (Java, Python, etc.) to integrate DynamoDB into applications programmatically.

**Summary of today’s learning**:

* Understood **AWS CLI** installation & usage basics.
* Explored **Amazon DynamoDB** – its features, advantages, and differences from SQL.
* Learned **core operations** on tables and items.
* Covered **CAP theorem relevance** to DynamoDB.
* Saw tools like **NoSQL Workbench** (GUI) and **DynamoDB Local** (for local testing).