# **Day 31 - 6 September 2025**

**Document Name:**Day 31 - hmuvvala@ - Hari Gopal Muvvala

### **1. DynamoDB Architecture**

* **Tables** → core containers for data (similar to RDBMS tables).
* **Items** → individual records stored in a table.
* **Attributes** → fields within an item.
* **Primary key** → uniquely identifies each item (Partition Key or Partition + Sort Key).
* **Partitions** → data is distributed across partitions for scalability.
* **Provisioned throughput** → read and write capacity units allocated per table.
* **Indexes** → support flexible querying (Local Secondary Index, Global Secondary Index).

### **2. Design Patterns (context of DynamoDB)**

* **Repository pattern** → isolate database operations behind a repository class.
* **Singleton pattern** → maintain a single DynamoDB client object across the application.
* **Factory pattern** → create DynamoDB client instances or objects with encapsulated configuration.
* These patterns improve **reusability, maintainability, and abstraction**.

### **3. EC2 and its Features**

* **Elastic Compute Cloud (EC2)** is a virtual server hosting service by AWS.
* Key features:
  + On-demand resizable compute capacity.
  + Different instance types for general purpose, compute-optimized, memory-optimized, etc.
  + Auto Scaling to handle variable workloads.
  + Security via VPC, Security Groups, IAM roles.
  + Flexible pricing models: On-demand, Reserved, Spot Instances.
  + Integration with EBS (Elastic Block Storage) and S3.

### **4. CAP Theorem**

* **Consistency** → all nodes see the same data at the same time.
* **Availability** → every request gets a response (even if not the most recent).
* **Partition Tolerance** → system continues to operate despite network splits.
* A distributed system can fully guarantee only **two** of the three.
* DynamoDB emphasizes **Availability** and **Partition Tolerance**, while offering configurable consistency (eventual or strong consistency).

### **5. ACID vs BASE Properties**

* **ACID (traditional databases)**
  + Atomicity → operations complete fully or not at all.
  + Consistency → valid state transitions.
  + Isolation → transactions don’t interfere with each other.
  + Durability → committed data persists.
* **BASE (NoSQL approach)**
  + Basically Available → system guarantees availability.
  + Soft-state → state may change over time, even without input.
  + Eventual consistency → system will become consistent over time.
* DynamoDB balances both: supports **ACID transactions** when needed, but often operates with **BASE principles**for scalability.