# Databases in AWS

## Choosing the Right Database

- AWS offers many managed databases to choose from.
- To select the appropriate database, consider the following questions:
    - Is the workload read-heavy, write-heavy, or balanced?
    - What are the throughput needs? Will they fluctuate or need to scale during the day?
    - How much data do you need to store, and for how long?
    - Will the data volume grow? What is the average object size? How will the data be accessed?
    - What level of data durability is required? Is this the source of truth for your data?
    - What are the latency requirements? How many concurrent users will there be?
    - What is the data model? How will the data be queried? Are joins needed? Is the data structured or semi-structured?
    - Do you need a strong schema or more flexibility? Will the data be used for reporting or search?
    - Would a traditional RDBMS or a NoSQL solution be more appropriate?
    - Are there license costs? Can you switch to a Cloud Native DB like Aurora?

## Database Types

- **RDBMS (SQL / OLTP):**
    - Amazon RDS, Amazon Aurora
    - Best suited for relational data and complex joins
- **NoSQL Databases:**
    - No joins, no SQL
    - Examples:
        - DynamoDB (~JSON structure)
        - ElastiCache (key/value store)
        - Neptune (graph database)
        - DocumentDB (compatible with MongoDB)
        - Keyspaces (compatible with Apache Cassandra)
- **Object Store:**
    - Amazon S3 (for storing large objects)
    - Amazon Glacier (for long-term backups and archives)
- **Data Warehouse (SQL Analytics / BI):**
    - Amazon Redshift (OLAP)
    - Amazon Athena
    - Amazon EMR
- **Search Engines:**
    - Amazon OpenSearch
    - Suitable for free text and unstructured data searches
- **Graph Databases:**

    - Amazon Neptune
    - Designed for displaying relationships between data

- **Ledger Databases:**

    - Amazon Quantum Ledger Database (QLDB)

- **Time Series Databases:**

    - Amazon Timestream

> *Note: Some of these databases are covered in more detail in the Data & Analytics section.*

## Amazon RDS – Summary

- Managed relational databases:

    - PostgreSQL, MySQL, Oracle, SQL Server, DB2, MariaDB
    - RDS Custom for Oracle & SQL Server (access to underlying instance)

- Configuration options:

    - Provisioned instance size
    - EBS volume type and size
    - Auto-scaling storage

- High availability and scalability:

    - Read Replicas
    - Multi-AZ deployments

- Security:

    - IAM integration
    - Security Groups
    - KMS (encryption at rest)
    - SSL (encryption in transit)

- Backup & recovery:

    - Automated backups (point-in-time restore, up to 35 days)
    - Manual DB snapshots for long-term backup

- Maintenance:

    - Managed and scheduled (with possible downtime)

- Authentication:

    - IAM authentication support
    - Integration with AWS Secrets Manager

### Use Case:

- Store relational datasets (RDBMS / OLTP)
- Execute SQL queries and transactions

## Amazon Aurora – Summary

- **Compatibility:**
  - API compatible with PostgreSQL and MySQL
  - Separation of storage and compute
- **Storage:**
  - Data stored in 6 replicas across 3 Availability Zones
  - Highly available, self-healing, auto-scaling
- **Compute:**
  - Cluster of DB instances across multiple AZs
  - Auto-scaling of read replicas
- **Cluster Architecture:**
  - Custom endpoints for writer and reader instances
- **Features:**
  - Same security, monitoring, and maintenance as Amazon RDS
  - Backup and restore options similar to RDS
  - Aurora Serverless: ideal for unpredictable or intermittent workloads, no capacity planning
  - Aurora Global: up to 16 read replicas per region, with storage replication < 1 second
  - Aurora Machine Learning: integrate with SageMaker and Comprehend
  - Aurora Database Cloning: create a new cluster faster than restoring a snapshot

### Use Case:

- Same as Amazon RDS, but offers:
  - Less maintenance
  - More flexibility
  - Higher performance
  - Additional features

## Amazon ElastiCache – Summary

- **Supported Engines:**
  - Managed Redis and Memcached
  - Similar to RDS but designed for caching
- **Performance:**
  - In-memory data store with sub-millisecond latency
- **Configuration:**
  - Choose an ElastiCache instance type (e.g., `cache.m6g.large` )
  - Redis supports clustering and sharding
  - Multi-AZ support with read replicas
- **Security:**
  - IAM integration
  - Security Groups
  - KMS encryption

    - Redis Auth
- **Data Management:**
    - Backup and snapshot support
    - Point-in-time restore
- **Maintenance:**
    - Managed and scheduled (may involve downtime)
- **Integration:**
    - Requires changes in application code to make use of caching

### Use Case:

- Key/value store
- Frequent reads, fewer writes
- Cache results of database queries
- Store session data for web applications
- No SQL support

## Amazon DynamoDB – Summary

- **Overview:**
    - AWS proprietary, fully managed serverless NoSQL database
    - Millisecond latency
- **Capacity Modes:**
    - Provisioned capacity (with optional auto-scaling)
    - On-demand capacity
- **Performance & Availability:**
    - Highly available, multi-AZ by default
    - Decoupled read and write operations
    - Supports transactions
    - DAX (DynamoDB Accelerator) cluster for caching, microsecond read latency
- **Use as Key/Value Store:**
    - Can replace ElastiCache (e.g., for session data)
    - Supports TTL (Time To Live) for automatic record expiration
- **Security:**
    - IAM-based authentication and authorization
- **Event Integration:**
    - DynamoDB Streams for triggering AWS Lambda or integrating with Kinesis Data Streams
- **Global Tables:**
    - Active-active replication across regions
- **Backup and Restore:**

  - Automated backups with Point-In-Time Restore (PITR) – up to 35 days
  - On-demand backups
  - Export to S3 without consuming read capacity units (RCU)
  - Import from S3 without consuming write capacity units (WCU)
- **Schema Design:**
  - Excellent for applications that require rapid schema evolution

### Use Case:

- Serverless application development
- Distributed serverless cache
- Suitable for storing small documents (hundreds of KB)

## Amazon S3 – Summary

- **Storage Model:**
  - Key/value store for objects
  - Optimized for large objects (not ideal for many small ones)
- **Scalability & Limits:**
  - Serverless and infinitely scalable
  - Maximum object size: 5 TB
  - Supports versioning
- **Storage Classes (Tiers):**
  - S3 Standard
  - S3 Infrequent Access
  - S3 Intelligent Tiering
  - S3 Glacier (for archival)
  - Lifecycle policies can transition objects between tiers
- **Features:**
  - Versioning
  - Encryption
  - Replication
  - MFA-Delete
  - Access logs
- **Security:**
  - IAM policies
  - Bucket policies
  - ACLs (Access Control Lists)
  - Access Points
  - Object Lambda
  - CORS support
  - Object Lock / Vault Lock
- **Encryption Options:**
  - SSE-S3 (managed by S3)

    - SSE-KMS (managed by KMS)
    - SSE-C (customer-provided keys)
    - Client-side encryption
    - TLS for encryption in transit
    - Default encryption setting
- **Operations & Automation:**
    - Batch operations with S3 Batch
    - File listing with S3 Inventory
    - Multi-part uploads
    - S3 Transfer Acceleration (improve upload/download speed)
    - S3 Select (query partial content)
    - Event Notifications via SNS, SQS, Lambda, EventBridge

### Use Cases:

- Static file storage
- Key/value store for large files
- Static website hosting

## Amazon DocumentDB – Summary

- **Purpose:**
    - AWS's implementation of MongoDB (just like Aurora is for PostgreSQL/MySQL)
    - Designed for storing, querying, and indexing JSON data
- **Architecture:**
    - Similar deployment concepts as Amazon Aurora
    - Fully managed and highly available
    - Replication across 3 Availability Zones
- **Storage:**
    - Automatically grows in 10 GB increments
    - Scales to support workloads with millions of requests per second

### Use Case:

- Applications that use MongoDB-like data models
- NoSQL storage with JSON document support

## Amazon Neptune – Summary

- **Overview:**
    - Fully managed graph database service
- **Example Use Case:**
    - Social network data model:
        - Users have friends
        - Posts have comments
        - Comments have likes from users

        - Users share and like posts

- **Performance & Scalability:**
    - Optimized for highly connected datasets
    - Handles complex and difficult graph queries efficiently
    - Millisecond latency even with billions of relationships
- **Availability:**
    - Replication across 3 Availability Zones
    - Up to 15 read replicas

### Ideal Use Cases:

- Knowledge graphs (e.g., Wikipedia)
- Fraud detection
- Recommendation engines
- Social networking platforms

## Amazon Neptune – Streams

- Provides a **real-time ordered sequence** of every change to your graph data
- Changes are available **immediately** after writing
- Ensures **no duplicates** and **strict order**
- Stream data is accessible via an **HTTP REST API**

### Use Cases:

- Send notifications when specific changes occur in the graph
- Keep your graph data synchronized in another data store:
    - Amazon S3
    - Amazon OpenSearch
    - Amazon ElastiCache
- Replicate graph data across regions in Neptune

### Integration Flow:

- Neptune Cluster writes data
- Neptune Streams expose changes via REST API
- A streams reader application fetches updates and forwards them to target systems

## Amazon Keyspaces (for Apache Cassandra)

- **Overview:**
    - Managed Apache Cassandra–compatible database service
    - Based on open-source Apache Cassandra (NoSQL, distributed)
- **Features:**
    - Serverless and scalable
    - Fully managed and highly available
    - Automatically scales tables up or down based on traffic

  - Tables are replicated 3 times across multiple Availability Zones
  - Uses Cassandra Query Language (CQL)
  - Single-digit millisecond latency at any scale
  - Supports thousands of requests per second

- **Capacity Modes:**

  - On-demand mode
  - Provisioned mode with auto-scaling

- **Data Protection:**

  - Encryption
  - Backups
  - Point-In-Time Recovery (PITR) – up to 35 days

### Use Cases:

- Storing IoT device information
- Time-series data

## Amazon QLDB – Summary

- **What is QLDB?**

  - QLDB stands for "Quantum Ledger Database"
  - A ledger is a system for recording financial transactions

- **Features:**

  - Fully managed and serverless
  - Highly available with replication across 3 Availability Zones
  - Designed to track and review the complete history of data changes
  - **Immutable:** entries cannot be deleted or modified
  - **Cryptographically verifiable** history of changes

- **Performance:**

  - 2–3x better performance than common blockchain ledger frameworks
  - Supports SQL-like language for data manipulation

- **Comparison with Amazon Managed Blockchain:**

  - QLDB is **not decentralized**
  - Aligned with **financial regulation compliance**

### Use Cases:

- Applications that require an immutable, verifiable history of changes
- Financial transaction tracking

## Amazon Timestream – Summary

- **Overview:**

  - Fully managed, fast, scalable, and serverless time series database
  - Automatically scales up or down based on demand

    - Capable of storing and analyzing trillions of events per day

- **Performance & Cost:**
    - Thousands of times faster than traditional relational databases
    - Operates at 1/10th the cost
- **Features:**
    - Scheduled queries
    - Multi-measure records
    - SQL-compatible querying
    - Built-in time series analytics functions for near real-time pattern detection
- **Storage Architecture:**
    - Recent data is kept in memory
    - Historical data is moved to a cost-optimized tier
- **Security:**
    - Encryption in transit and at rest

### Use Cases:

- Internet of Things (IoT) applications
- Operational monitoring and metrics
- Real-time analytics

## Amazon Timestream – Architecture

Amazon Timestream integrates seamlessly with various AWS services and external tools to collect, process, analyze, and visualize time series data.

### Data Ingestion Sources:

- **AWS IoT:** Collects data from IoT devices.
- **Kinesis Data Streams:** Streams large volumes of data in real time.
- **Amazon MSK (Managed Streaming for Apache Kafka):** For event-driven architectures.
- **Lambda:** Used for serverless processing and transformation of time series data.

### Data Processing and Analytics:

- **Kinesis Data Analytics for Apache Flink:** Performs advanced analytics and stream processing.
- **Prometheus:** Can be integrated for collecting monitoring metrics.

### Data Access and Visualization:

- **Amazon QuickSight:** For dashboarding and visualization of time series trends.
- **Amazon SageMaker:** For applying machine learning on historical time series data.
- **JDBC Connections:** External tools can connect via standard SQL-based interfaces.

Timestream acts as the central time series data store in this architecture.