

# Picking the Right Databases

## How to pick right database

Depend on the question, read-heavy, write-heavy, or balanced workload? throughput needs?

How much data to store and for how long? Will it grow? Data durability?

Latency requirements?

Strong schema or flexibility? It depends on a lot of factors

## RDS

Have fully managed postgresSQL / MySQL / Oracle / SQL Server / Maria DB

You will need to provision the RDS instance size and EBS volume type, but you can set up auto-scaling for storage

Read replicas for multi-AZ to give it more read capability or for another application so that it doesn't affect production access.

**Use case:** For storing relational databases OLTP online transactional processing like purchasing

## Aurora

Compatible with Postgres and MySQL. It is AWS's proprietary database.

Data are stored in 6 replicas, across 3 AZ, highly available and have self-healing, also auto-scaling out of the box.

You can define custom endpoints for writer and reader database instances.

**Aurora serverless:** For unpredictable / intermittent workloads, no capacity planning

**Aurora Multi-Masster:** For fast failover for writers

**Aurora Global:** Up to 16 database read instance in each region

Cloning Aurora is much faster than snapshot restoring.

# ElastiCache

In-memory data store that give you sub-millisecond latency. You need to provision EC2 instance type.

Redis give you multi-AZ and read replicas, and it is preserved when shutting down.

However, using ElastiCache you need to modify your code heavily.

ElastiCache is good for user session store.

# DynamoDB

Fully managed serverless NoSQL database, millisecond latency. You can migrate MongoDB to here because it is key based as well with primary key.

Provisioned capacity: Used for smooth workload

On-demand capacity: Used for unpredictable workload

DynamoDB can replace ElastiCache as key/value store, can automatically expire data.

Highly available, multi-AZ, you can scale read/write independently.

**DAX accelerator for microsecond read latency.**

You can enable stream to react to database changes, insertion, update, delete and make it invoke lambda functions.

Good for schemas that are constantly changing.

# S3

Key / value store, good for storing big objects. Max object size is 5 TB! Can have version capability.

# DocumentDB

Aurora version of MongoDB, it is NoSQL database.

Used to store, query, and index JSON data.

Scales automatically, data replicated across 3 AZ.

# Neptune

Fully managed graph database. Have edges and nodes.

## Keyspaces

Managed Apache Cassandra. Serverless, scalable, highly available.

Another NoSQL distributed databases.

## QLDB

Quantum ledger database. Ledger is a book recording financial transactions.

Again is highly available, serverless.

Review history of all changes made to your application. Immutable and cryptographically verifiable, basically the blockchain.

No decentralization, it is a central database

## Timestream

Time + value. Basically a time series database.

Store and analyze trillions of events per day.

It is very fast if you are using time based data.

---

Revision #2

Created 2023-02-24 19:00:09 UTC by Tamarine

Updated 2023-02-24 19:34:39 UTC by Tamarine