Database services on AWS

Introduction

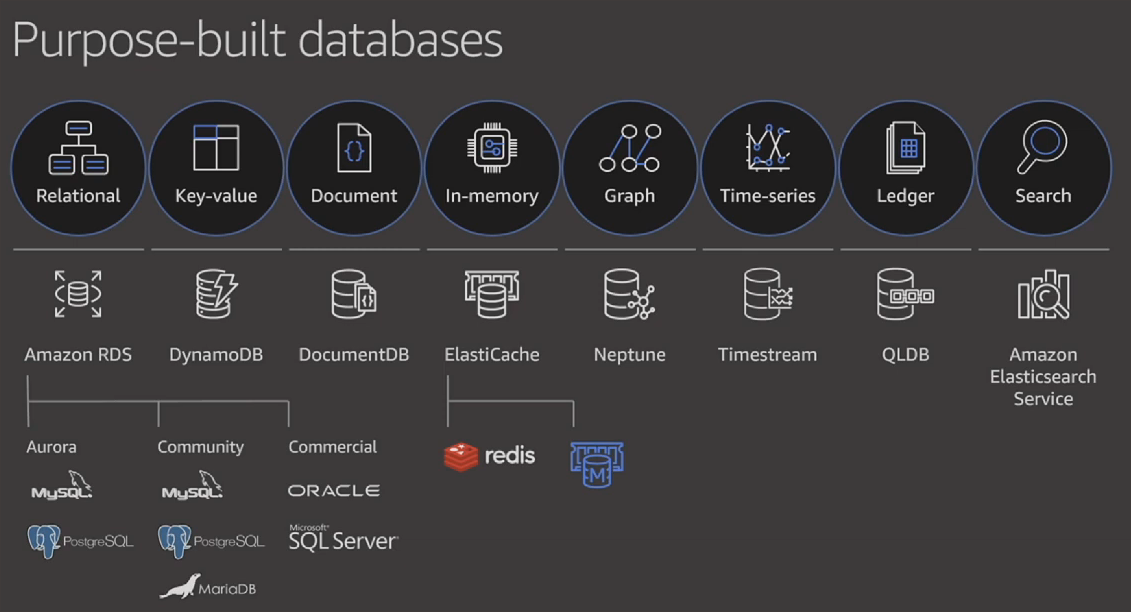
AWS provides a large collection of database services, which different organizations can utilize for their needs. Here’s a short list of features, which make database services on AWS stand out:

1. Purpose-built – Choose from 15+ purpose-built database engines including relational, key-value, document, in-memory, graph, time series, wide column, and ledger databases. AWS’s portfolio of purpose-built databases supports diverse data models and allows you to build use case driven, highly scalable, distributed applications.
2. Performance at scale – Get relational databases that are 3-5X faster than popular alternatives, or non-relational databases that give you microsecond to sub-millisecond latency. Start small and scale as your applications grow. You can scale your database's compute and storage resources easily, often with no downtime.
3. Fully managed – With AWS databases, you don’t need to worry about database management tasks such as server provisioning, patching, configuration, or backups. AWS continuously monitors your clusters to keep your workloads running with self-healing storage and automated scaling, so that you can focus on application development.
4. Secure & highly available – AWS databases are built for business-critical, enterprise workloads, offering high availability, reliability, and security. These databases support multi-region, multi-master replication, and provide full oversight of your data with multiple levels of security, including network isolation, and end-to-end encryption.

AWS Databases cover a large number of possible use-cases, however will go over a selection of most popular use cases, that allow you to use full power of AWS database services. These uses cases include:

* Amazon Relational Database Service
* Amazon DynamoDB
* Amazon ElastiCache

Below you can find a picture, which gives you a brief overview of possible ways to utilize different database services.



Amazon Relational Database Service

*Basically, RDS is a SQL database managed by Amazon. So usecases are the same as SQL DB’s usecases (e.g., situations where data integrity is absolutely paramount, like financial applications, defense and security, private health information, or highly structured data).*

**General Info**

[Amazon Relational Database Service](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html) (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

**General Info**

Amazon RDS takes over many of the difficult and tedious management tasks of a relational database:

* When you buy a server, you get CPU, memory, storage, and IOPS, all bundled together. With Amazon RDS, these are split apart so that you can scale them independently. If you need more CPU, less IOPS, or more storage, you can easily allocate them.
* You can use the [database products](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html#Welcome.Concepts.DBInstance) you are already familiar with: MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server.
* You can get [high availability](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.MultiAZ.html) with a primary instance and a synchronous secondary instance that you can fail over to when problems occur. You can also use MariaDB, Microsoft SQL Server, MySQL, Oracle, and PostgreSQL read replicas to increase read scaling.
* In addition to the security in your database package, you can help control who can access your RDS databases by using [AWS Identity and Access Management](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/security_iam_service-with-iam.html) (IAM) to define users and permissions. You can also help protect your databases by putting them in a virtual private cloud.

**Governance**

* Amazon RDS manages backups, software patching, automatic failure detection, and recovery.
* To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances. It also restricts access to certain system procedures and tables that require advanced privileges.
* You can have automated [backups](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_WorkingWithAutomatedBackups.html) performed when you need them, or manually create your own backup snapshot. You can use these backups to restore a database. The Amazon RDS restore process works reliably and efficiently.

**Constrains/Cautions**

* By default, you can have up to a total of 40 DB instances. RDS DB instances, Aurora DB instances, Amazon Neptune instances, and Amazon DocumentDB instances apply to this quota.
* The maximum number of simultaneous database connections varies by the DB engine type and the memory allocation for the DB instance class. Refer to the [documentation](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Limits.html).
* There are some naming constraints in Amazon RDS. Refer to the [documentation](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Limits.html).

**Pricing**

[*Link to cost estimation of HA RDS deployment with 3 instances with 100 GB (about ~370$ per month)*](https://calculator.aws/#/estimate?id=0452daa80baacaac5a6365e8ab437f69bd16cc92)

Amazon DynamoDB

*Whenever SQL database (like RDS) is not suitable for your needs (e.g., data structure is not the same across all items), DynamoDB is a great choice for such cases.*

***General info***

[Amazon DynamoDB](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html) is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.

* In DynamoDB, tables, items, and attributes are the [core components](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.CoreComponents.html) that you work with. A *table* is a collection of *items*, and each item is a collection of *attributes*.
* DynamoDB supports [eventually consistent and strongly consistent reads](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadConsistency.html).
* DynamoDB comes in two [Read/Write capacity modes](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html), which should be chosen depending on application load and your budget.
* For working with data, you can access Amazon DynamoDB using the AWS Management Console, the AWS Command Line Interface (AWS CLI), or the DynamoDB API.
* If your application requires [really fast reads](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DAX.html#DAX.use-cases) (real-time bidding, social gaming, and trading applications) [DynamoDB Accelerator (DAX)](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DAX.html) is a good choice, as it will cache query data from DynamoDB and your application will get the needed data much faster.
* As DynamoDB has two Read/Write capacity modes, it’s important to understand, whether to use [On-Demand](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html#HowItWorks.OnDemand) or [Provisioned](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html#HowItWorks.ProvisionedThroughput.Manual) mode.

**Governance**

* DynamoDB provides on-demand backup capability. It allows you to create full backups of your tables for long-term retention and archival for regulatory compliance needs. For more information, see [On-Demand Backup and Restore for DynamoDB](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/BackupRestore.html).
* You can create on-demand backups and enable point-in-time recovery for your Amazon DynamoDB tables. Point-in-time recovery helps protect your tables from accidental write or delete operations. With point-in-time recovery, you can restore a table to any point in time during the last 35 days. For more information, see [Point-in-Time Recovery: How It Works](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/PointInTimeRecovery_Howitworks.html).
* DynamoDB allows you to delete expired items from tables automatically to help you reduce storage usage and the cost of storing data that is no longer relevant. For more information, see [Expiring Items By Using DynamoDB Time to Live (TTL)](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/TTL.html).
* Dynamo is also integrated with CloudWatch, allowing you to enable [logging and monitoring](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/MonitoringDynamoDB.html).

**Cautions/Constrains**

Dynamo has a [number of quotas](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Limits.html) you need to pay attention to.

One of the most important limits is [Read/Write throughput](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Limits.html#default-limits-throughput-capacity-modes). If you exceed this limit, all queries and requests will be throttled. Setting up CloudWatch alerts to track such cases would be really nice.

**Pricing considerations**

All info regarding DynamoDB pricing can be found in [AWS docs](https://aws.amazon.com/dynamodb/pricing/)

Amazon ElastiCache

*Amazon ElastiCache allows you to seamlessly set up, run, and scale popular open-source compatible in-memory data stores in the cloud. Build data-intensive apps or boost the performance of your existing databases by retrieving data from high throughput and low latency in-memory data stores. Amazon ElastiCache is a popular choice for real-time use cases like Caching, Session Stores, Gaming, Geospatial Services, Real-Time Analytics, and Queuing.*

**General Info**

Amazon ElastiCache makes it easy to set up, manage, and scale distributed in-memory cache environments in the AWS Cloud. It provides a high performance, resizable, and cost-effective in-memory cache, while removing complexity associated with deploying and managing a distributed cache environment. ElastiCache works with both the Redis and Memcached engines.

* [ElastiCache is fully integrated with Amazon VPC](https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/VPCs.EC.html), using the same principles, which are used for EC2 VMs.

**ElastiCache Redis**

* The basic building block of ElastiCache for Redis is the [cluster](https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/WhatIs.html#WhatIs.Clusters). A cluster is a collection of one or more cache nodes, all of which run an instance of the Redis cache engine software.
* A [node](https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/WhatIs.Components.html#WhatIs.Components.Nodes)is the smallest building block of an Redis deployment. A node can exist in isolation form or in some relationship to other nodes. A node is a fixed-size chunk of secure, network-attached RAM. Each node runs an instance of the engine and version that was chosen when you created your cluster.
* [Replication](https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/WhatIs.Components.html#WhatIs.Components.ReplicationGroups) is implemented by grouping from two to six nodes in a [shard](https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/WhatIs.Components.html#WhatIs.Components.Shards) (in the API and CLI, called a node group). One of these nodes is the read/write primary node. All the other nodes are read-only replica nodes.
* ElasticCache Redis supports a number of complex data objects, such as lists, hashes, sets, etc.
* ElastiCache for Redis supports (similar to RDS features)
  + Redis Master/Slave replication.
  + Multi-AZ operation by creating read replicas in another AZ
  + Backup and Restore feature for persistence using snapshots

**ElastiCache Memcached**

* A [node](https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/WhatIs.Components.html#WhatIs.Components.Nodes)is the smallest and basic building block of ElastiCache for Memcached. A node can exist in isolation form or in some relationship to other nodes. A node is a fixed-size chunk of secure, network-attached RAM. Each node runs an instance of the engine and version that was chosen when you created your cluster.
* A [Memcached cluster](https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/WhatIs.Components.html#WhatIs.Components.Clusters)is a logical grouping of one or more ElastiCache nodes. Data is partitioned across the nodes in a Memcached cluster.
* ElasticCache Redis supports simple data objects, such as strings, integers, etc.
* can be scaled Vertically by increasing the node type size
* can be scaled Horizontally by adding and removing nodes
* does not support persistence of data
* Memcached cluster can have
  + nodes can span across multiple AZs within the same region
  + maximum of 20 nodes per cluster with a maximum of 100 nodes per region (soft limit and can be extended)

**Governance**

* As most of AWS services, ElastiCache is fully integrated with [AWS CloudWatch](https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/CacheMetrics.html) for metrics, [AWS CloudTrail](https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/logging-using-cloudtrail.html) for logging and it also supports [AWS SNS](https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/ECEvents.html), sending all important events to a specific topic.
* For security, ElastiCache provides [several types of encryption](https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/encryption.html) and [Role Based Access Control](https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/Clusters.RBAC.html).

**Cautions/ Considerations**

* One of the most important choices in working with ElastiCache is whether to use Redis or Memcached engine. You can use [this cheetsheet](https://d0.awsstatic.com/whitepapers/performance-at-scale-with-amazon-elasticache.pdf#page=8&zoom=100,92,493) to decide what suits your needs better.
* You also might want to take a look at [reserved nodes](https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/CacheNodes.Reserved.html) as a way to reduce costs, if you can predict what kind of load will you have.