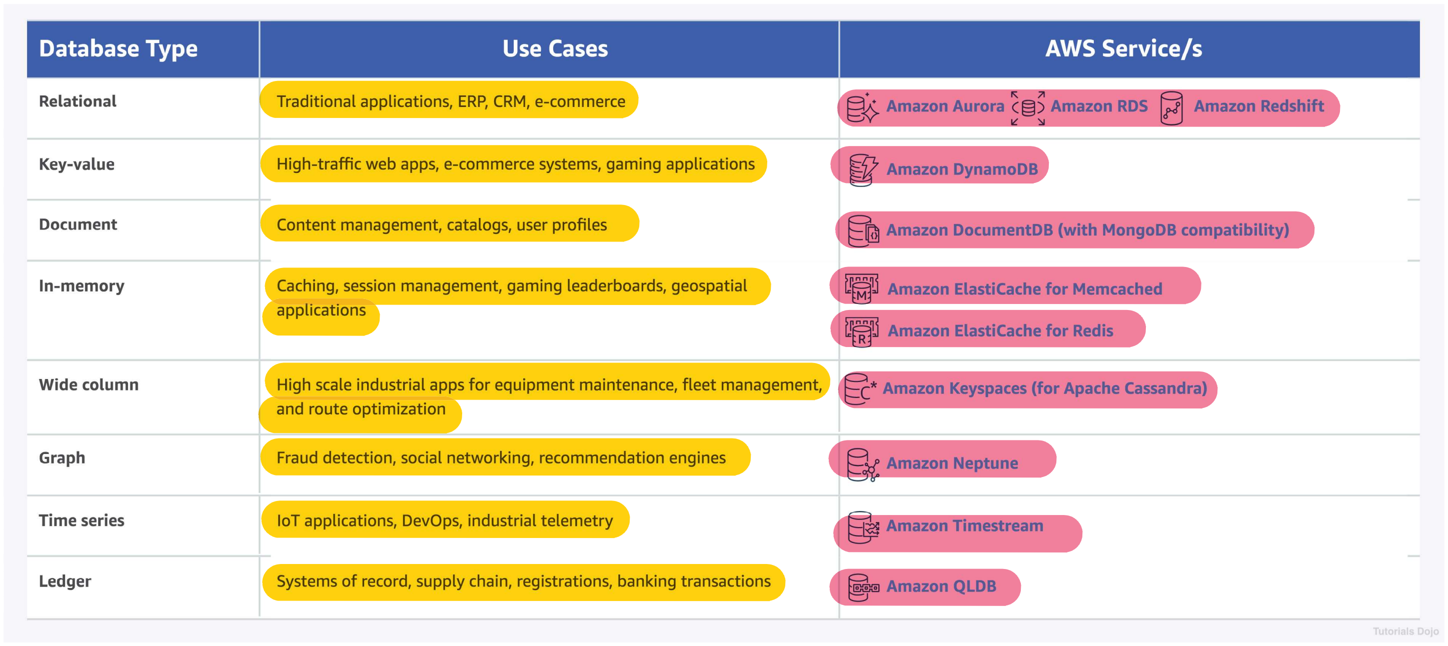
DATABASE

AWS offers purpose-built databases for all your application needs. Whether you need a Relational, Key-Value, In-memory, or any other type of data store, AWS would most likely have a database service that you can use.

Relational databases store data with predefined schemas and "relationships" between the tables, hence the "Relational" name. It is designed to support ACID (Atomicity, Consistency, Isolation, Durability) transactions with strong data consistency to maintain referential integrity. Key-value databases are suitable for storing and retrieving large volumes of data. It delivers quick response times even in large volumes of concurrent requests.

In-memory databases are primarily used for applications that require real-time access to data. It is capable of delivering data to applications in microseconds and not just in milliseconds since the data are directly stored in memory and not on disk. Aside from this, AWS also offers Document, Time Series, Ledger, and many other database types.



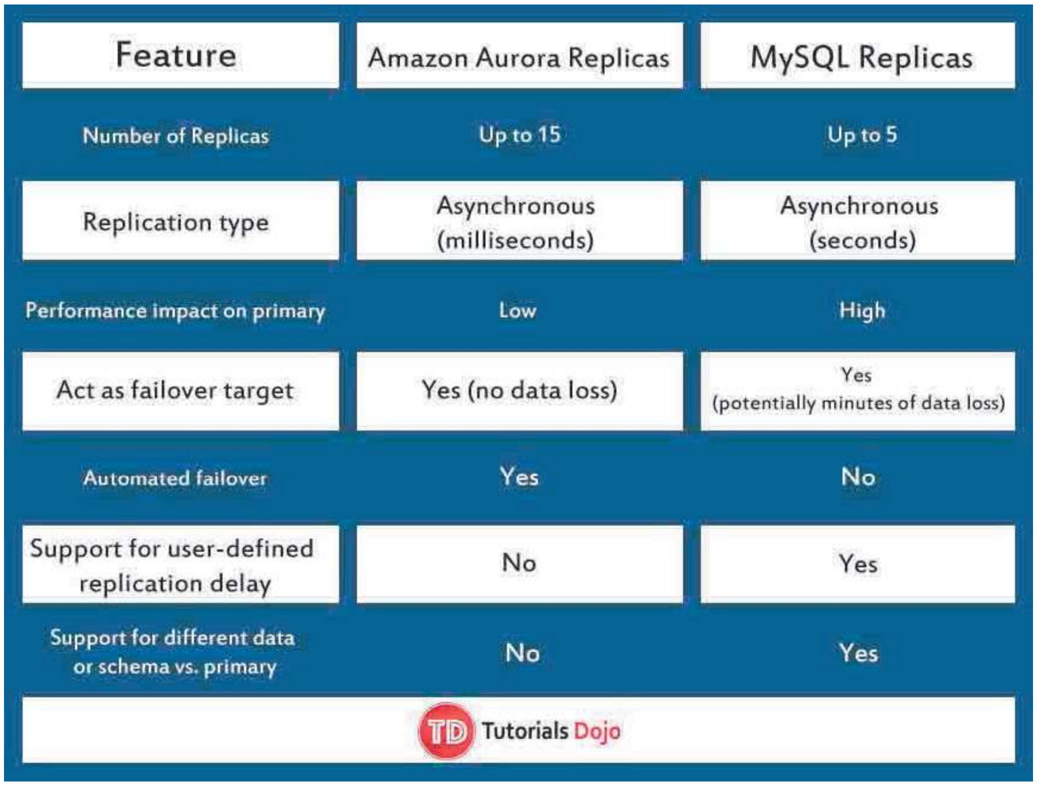
Database Type	Use Cases	AWS Service/s
Relational	Traditional applications, ERP, CRM, e-commerce	Amazon Aurora, Amazon RDS, Amazon Redshift
Key-value	High-traffic web apps, e-commerce systems, gaming applications	Amazon DynamoDB
Document	Content management, catalogs, user profiles	Amazon DocumentDB (with MongoDB compatibility)
In-memory	Caching, session management, gaming leaderboards, geospatial applications	Amazon ElastiCache for Memcached Amazon ElastiCache for Redis
Wide column	High scale industrial apps for equipment maintenance, fleet management, and route optimization	Amazon Keyspaces (for Apache Cassandra)
Graph	Fraud detection, social networking, recommendation engines	Amazon Neptune
Time series	IoT applications, DevOps, industrial telemetry	Amazon Timestream
Ledger	Systems of record, supply chain, registrations, banking transactions	Amazon QLDB

Amazon Aurora

- A fully managed relational database engine that's compatible with **MySQL** and **PostgreSQL**.
- Aurora includes a high-performance storage subsystem. The underlying storage grows automatically as needed, up to 128 terabytes.
- Aurora will keep your database up-to-date with the latest patches.
- Aurora is fault-tolerant and self-healing.
- Storage and Reliability
  - Aurora data is stored in the cluster volume, which is designed for reliability. A cluster volume consists of copies of the data across multiple Availability Zones in a single AWS Region.
  - Aurora automatically detects failures in the disk volumes that make up the cluster volume. When a segment of a disk volume fails, Aurora immediately repairs the segment. When Aurora repairs the disk segment, it uses the data in the other volumes that make up the cluster volume to ensure that the data in the repaired segment is current.
  - Aurora is designed to recover from a crash almost instantaneously and continue to serve your application data without the binary log. Aurora performs crash recovery asynchronously on parallel threads, so that your database is open and available immediately after a crash.
- High Availability and Fault Tolerance
  - When you create Aurora Replicas across Availability Zones, RDS automatically provisions and maintains them synchronously.
  - An Aurora DB cluster is fault tolerant by design. If the primary instance in a DB cluster fails, Aurora automatically fails over to a new primary instance in one of two ways:
    - By promoting an existing Aurora Replica to the new primary instance
    - By creating a new primary instance
  - Aurora storage is also self-healing. Data blocks and disks are continuously scanned for errors and repaired automatically.
  - Aurora backs up your cluster volume automatically and retains restore data for the length of the backup retention period, from 1 to 35 days.
  - Aurora automatically maintains **6 copies of your data across 3 Availability Zones** and will automatically attempt to recover your database in a healthy AZ with no data loss.
  - Aurora has a Backtrack feature that rewinds or restores the DB cluster to the time you specify. However, take note that the Amazon Aurora Backtrack feature is not a total replacement for fully backing up your DB cluster since the limit for a backtrack window is only 72 hours.
- Tags
  - You can use Amazon RDS tags to add metadata to your RDS resources.
  - Tags can be used with IAM policies to manage access and to control what actions can be applied to the RDS resources.
  - Tags can be used to track costs by grouping expenses for similarly tagged resources.
- Monitoring

- Subscribe to **Amazon RDS events** to be notified when changes occur with a DB instance, DB cluster, DB cluster snapshot, DB parameter group, or DB security group.
- Database log files
- Use CloudWatch Metrics, Alarms and Logs

- Security
    - Use IAM to control access.
    - To control which devices and EC2 instances can open connections to the endpoint and port of the DB instance for Aurora DB clusters in a VPC, you use a VPC security group.
    - You can make endpoint and port connections using Transport Layer Security (TLS) / Secure Sockets Layer (SSL). In addition, firewall rules can control whether devices running at your company can open connections to a DB instance.
    - Use RDS encryption to secure your RDS instances and snapshots at rest.



Feature	Amazon Aurora Replicas	MySQL Replicas
Number of Replicas	Up to 15	Up to 5
Replication type	Asynchronous (milliseconds)	Asynchronous (seconds)
Performance impact on primary	Low	High
Act as failover target	Yes (no data loss)	Yes (potentially minutes of data loss)
Automated failover	Yes	No
Support for user-defined replication delay	No	Yes
Support for different data or schema vs. primary	No	Yes

Tutorials Dojo

- Pricing
    - You are charged for DB instance hours, I/O requests, Backup storage and Data transfer.
    - You can purchase **On-Demand Instances** and pay by the hour for the DB instance hours that you use, or **Reserved Instances** to reserve a DB instance for a one-year or three-year term and receive a significant discount compared to the on-demand DB instance pricing.

Sources:

https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
https://aws.amazon.com/rds/aurora/serverless/
https://aws.amazon.com/rds/aurora/pricing/
https://aws.amazon.com/rds/aurora/faqs/

Amazon Relational Database Service (RDS)

- Industry-standard relational database
- RDS manages backups, software patching, automatic failure detection, and recovery.
- You can have automated backups performed when you need them, or manually create your own backup snapshot. You can use these backups to restore a database.
- Supports **Aurora**, **MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server**.
- Basic building block of RDS is the **DB instance**, which is an isolated database environment in the cloud.
- You can have up to 40 Amazon RDS DB instances.
- Each DB instance runs a **DB engine**.
- You can run your DB instance in several AZs, an option called a **Multi-AZ deployment**. Amazon automatically provisions and maintains a secondary standby DB instance in a different AZ. Your primary DB instance is synchronously replicated across AZs to the secondary instance to provide data redundancy, failover support, eliminate I/O freezes, and minimize latency spikes during system backups.
- DB Instance:
    - Endpoint: rds.<*region*>.amazonaws.com
    - Storage
        - Amazon RDS for MySQL, MariaDB, PostgreSQL, Oracle, and Microsoft SQL Server use Amazon EBS volumes for database and log storage.
        - Storage types :

          General Purpose SSD (gp2)
            - MySQL, MariaDB, Oracle, and PostgreSQL DB instances: 20 GiB–64 TiB storage size
            - SQL Server for Enterprise, Standard, Web, and Express editions: 20 GiB–16 TiB storage size

          Provisioned IOPS SSD (io1)

Database Engine	Range of Provisioned IOPS	Range of Storage
MariaDB	1,000–80,000	100 GiB–64 TiB
SQL Server, Enterprise and Standard editions	1000–32,000 or 64,000 for Nitro-based m5 instance types	20 GiB–16 TiB
SQL Server, Web and Express editions	1000–32,000 or 64,000 for Nitro-based m5 instance types	100 GiB–16 TiB
MySQL	1,000–80,000	100 GiB–64 TiB
Oracle	1,000–80,000	100 GiB–64 TiB

PostgreSQL	1,000–80,000	100 GiB–64 TiB

- For production OLTP use cases, use **Multi-AZ deployments** for enhanced fault tolerance with Provisioned IOPS storage for fast and predictable performance.
- Magnetic
    - Doesn't allow you to scale storage when using the SQL Server database engine.
    - Doesn't support elastic volumes.
    - Limited to a maximum size of 3 TiB.
    - Limited to a maximum of 1,000 IOPS.

Security

- Security Groups
    - **DB Security Groups** - controls access to a DB instance that is not in a VPC. By default, network access is turned off to a DB instance. This SG is for the EC2-Classic platform.
    - **VPC Security Groups** - controls access to a DB instance inside a VPC. This SG is for the EC2-VPC platform.
    - **EC2 Security Groups** - controls access to an EC2 instance and can be used with a DB instance.
- Practices
    - Assign an individual **IAM** account to each person who manages RDS resources. Do not use AWS root credentials to manage RDS resources.
    - Grant each user the minimum set of permissions required to perform his or her duties.
    - Use IAM groups to effectively manage permissions for multiple users.
    - Rotate your IAM credentials regularly.
    - Use **security groups** to control what IP addresses or Amazon EC2 instances can connect to your databases on a DB instance.
    - Run your DB instance in an Amazon Virtual Private Cloud (**VPC**) for the greatest possible network access control.
    - Use **Secure Socket Layer (SSL) connections** with DB instances running the MySQL, MariaDB, PostgreSQL, Oracle, or Microsoft SQL Server database engines.
    - Use RDS encryption to secure your RDS instances and snapshots at rest.
    - Use the security features of your DB engine to control who can log in to the databases on a DB instance.
- Encryption
    - At rest and in-transit.
    - Manage keys used for encrypted DB instances using the AWS KMS. KMS encryption keys are specific to the region that they are created in.

- RDS encryption is currently available for all database engines and storage types. RDS encryption is available for most DB instance classes.
- You can't restore an unencrypted backup or snapshot to an encrypted DB instance.
- You can use **SSL** from your application to encrypt a connection to a DB instance running MySQL, MariaDB, SQL Server, Oracle, or PostgreSQL.

- Amazon RDS supports the following scenarios for accessing a DB instance in a VPC:

DB Instance	Accessed By
In a VPC	An EC2 Instance in the Same VPC
	An EC2 Instance in a Different VPC
	An EC2 Instance Not in a VPC
	A Client Application Through the Internet
Not in a VPC	An EC2 Instance in a VPC
	An EC2 Instance Not in a VPC
	A Client Application Through the Internet

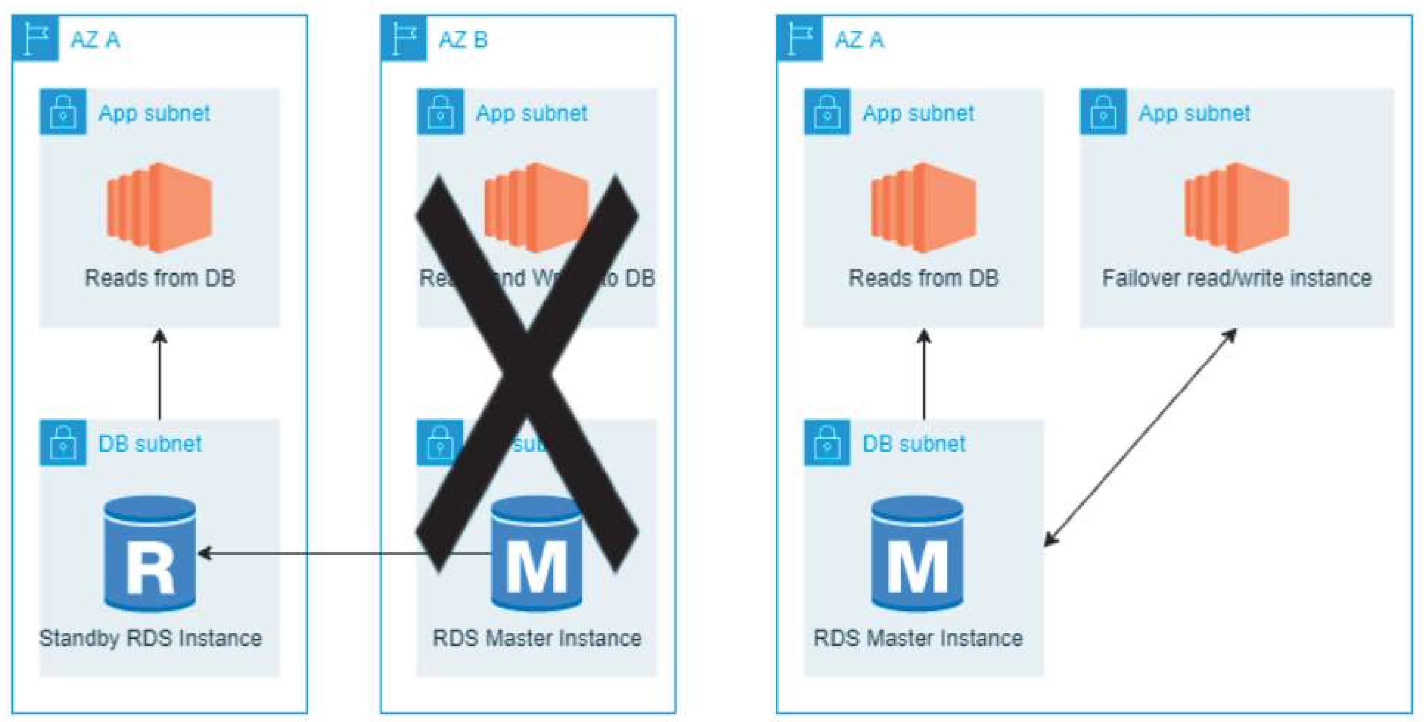
Tagging

- An RDS tag is a **name-value pair** that you define and associate with an RDS resource. The name is referred to as the key. Supplying a value for the key is optional.
- All Amazon RDS resources can be tagged.
- Use tags to organize your AWS bill to reflect your own cost structure.
- A *tag set* can contain as many as 50 tags, or it can be empty.

High Availability using Multi-AZ

- Multi-AZ deployments for **Oracle, PostgreSQL, MySQL, and MariaDB** DB instances use **Amazon's failover technology**. **SQL Server DB** instances use **SQL Server Mirroring**.
- **Amazon RDS for SQL Server** offers **Always On Availability Groups** for the Multi-AZ configuration in all AWS Regions. This is available for both Standard and Enterprise editions.
- You can modify a DB instance in a Single-AZ deployment to a Multi-AZ deployment.
- The primary DB instance switches over automatically to the standby replica if any of the following conditions occur:
  - An Availability Zone outage

- ○ The primary DB instance fails
- ○ The DB instance's server type is changed
- ○ The operating system of the DB instance is undergoing software patching
- ○ A manual failover of the DB instance was initiated using **Reboot with failover**

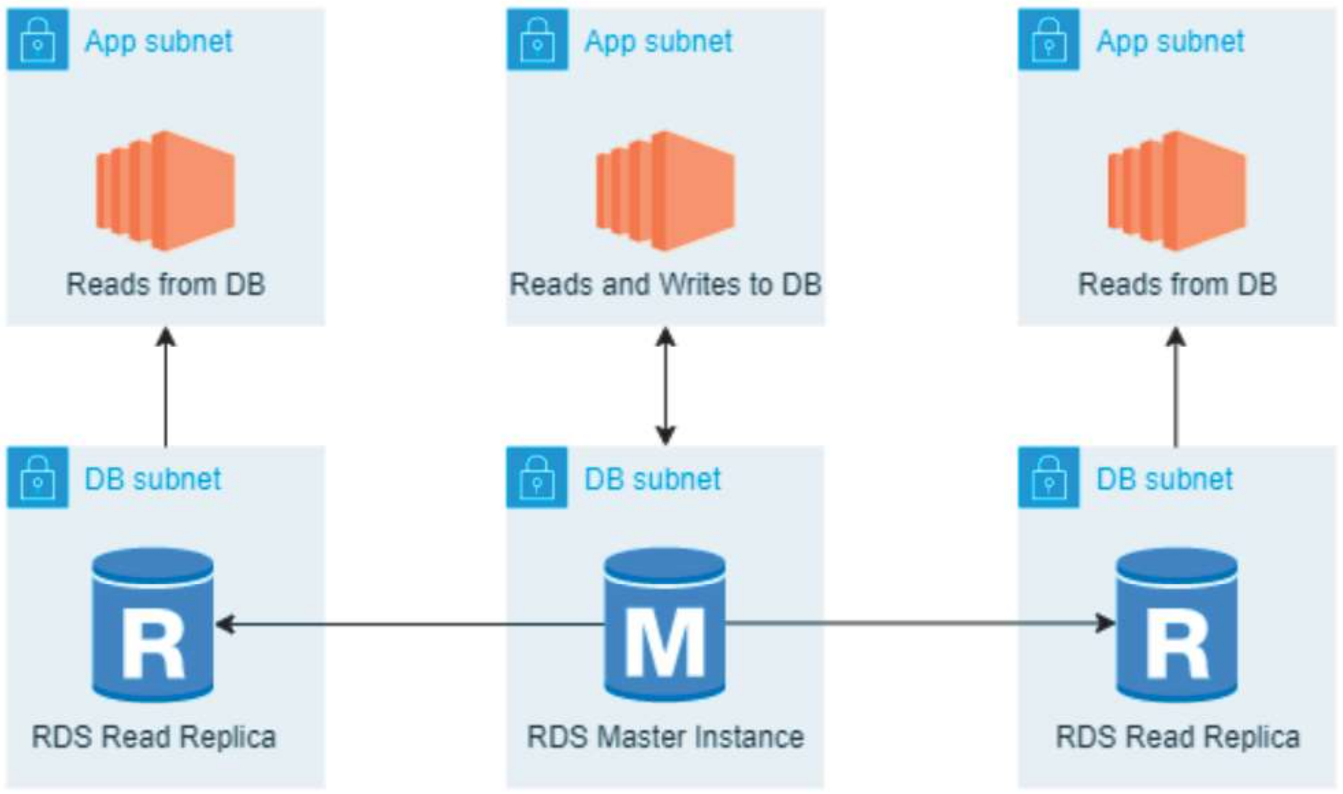


Standby DB instance in another subnet. AZ B experiences outage.

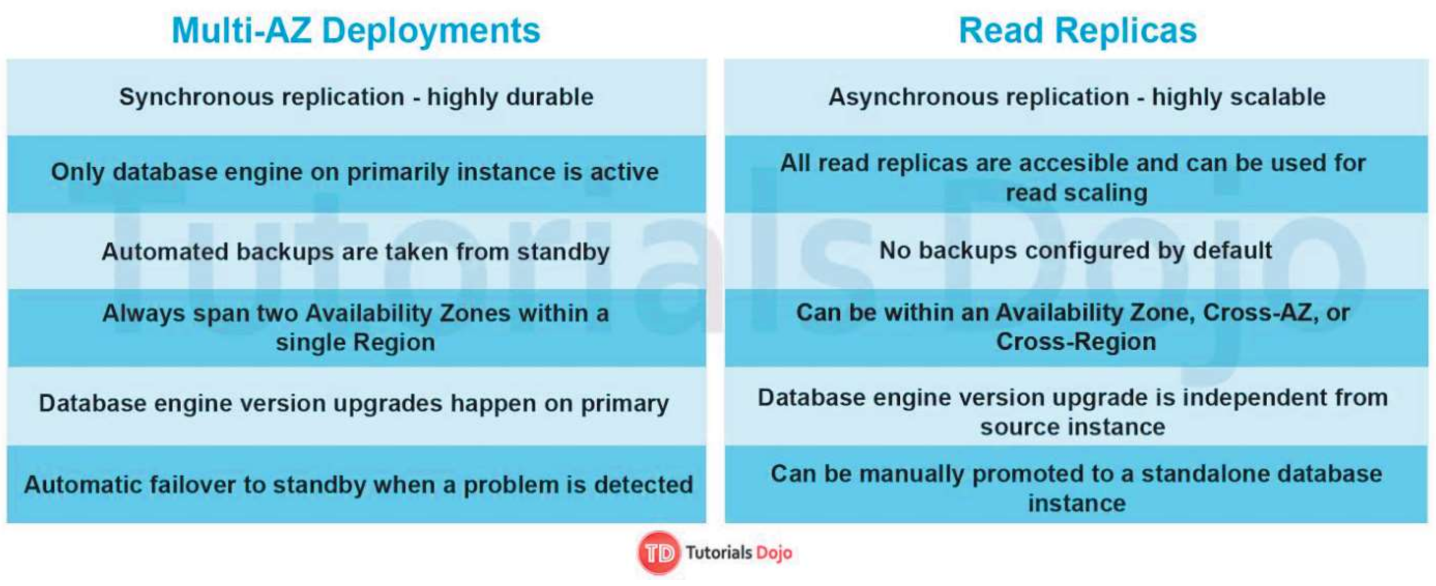
Read replica promoted to Master instance. Spawn a new read/write app in AZ A. Reroute traffic using failover policies.

Read Replicas

- Updates made to the source DB instance are asynchronously copied to the Read Replica.
- You can reduce the load on your source DB instance by routing read queries from your applications to the Read Replica.



Multi-AZ Deployments vs Read Replicas



Multi-AZ Deployments	Read Replicas
Synchronous replication - highly durable	Asynchronous replication - highly scalable
Only database engine on primarily instance is active	All read replicas are accesible and can be used for read scaling
Automated backups are taken from standby	No backups configured by default
Always span two Availability Zones within a single Region	Can be within an Availability Zone, Cross-AZ, or Cross-Region
Database engine version upgrades happen on primary	Database engine version upgrade is independent from source instance
Automatic failover to standby when a problem is detected	Can be manually promoted to a standalone database instance

Backups and Restores

- Your DB instance must be in the **ACTIVE state** for automated backups to occur.
- The first snapshot of a DB instance contains the data for the full DB instance. Subsequent snapshots of the same DB instance are incremental.

Monitoring

- Amazon CloudWatch
- RDS Events
  - An Amazon RDS event is created when the reboot is completed.
  - Be notified when changes occur with a DB instance, DB snapshot, DB parameter group, or DB security group.
  - Uses the Amazon Simple Notification Service (SNS) to provide notification when an Amazon RDS event occurs.
- Database log files
- CloudWatch gathers metrics about CPU utilization **from the hypervisor** for a DB instance, and Enhanced Monitoring gathers its metrics **from an agent** on the instance.
- Instance Status - indicates the health of the instance.
- CloudTrail captures all API calls for RDS as events.

Pricing

- With Amazon RDS, you pay only for the RDS instances that are active.
- The data transferred for cross-region replication incurs RDS data transfer charges.
- Instances are billed for DB instance hours (per second), Storage (per GiB per month), I/O requests (per 1 million requests per month), Provisioned IOPS (per IOPS per month), Backup storage (per GiB per month), and Data transfer (per GB).
  - Amazon RDS is billed in one-second increments for database instances and attached storage. Pricing is still listed on a per-hour basis, but bills are now calculated down to the second and show usage in decimal form. There is a 10 minute minimum charge when an instance is created, restored or started.
- RDS purchasing options:
  - **On-Demand Instances –** Pay by the hour for the DB instance hours that you use.
  - **Reserved Instances –** Reserve a DB instance for a one-year or three-year term and receive a significant discount compared to the on-demand DB instance pricing.
- Amazon RDS is now billed in one-second increments for database instances and attached storage. Pricing is still listed on a per-hour basis, but bills are now calculated down to the second and show usage in decimal form. There is a 10 minute minimum charge when an instance is created, restored or started.

Sources:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
https://aws.amazon.com/rds/features/
https://aws.amazon.com/rds/pricing/
https://aws.amazon.com/rds/faqs/

Amazon DynamoDB

- NoSQL database service that provides fast and predictable performance with seamless scalability.
- Offers encryption at rest.
- You can create database tables that can store and retrieve any amount of data, and serve any level of request traffic.
- You can scale up or scale down your tables' throughput capacity without downtime or performance degradation, and use the AWS Management Console to monitor resource utilization and performance metrics.
- Provides on-demand backup capability as well as enable point-in-time recovery for your DynamoDB tables.
- All of your data is stored in partitions, backed by solid state disks (SSDs) and automatically replicated across multiple AZs in an AWS region, providing built-in high availability and data durability.
- Transactions provide atomicity, consistency, isolation, and durability (ACID) in DynamoDB, helping you to maintain data correctness in your applications.

Tagging

- Tags can help you:
  - Quickly identify a resource based on the tags you've assigned to it.
  - See AWS bills broken down by tags.
- Maximum number of tags per resource: 50

On-Demand Backup and Restore

- You can use IAM to restrict DynamoDB backup and restore actions for some resources.
- All backup and restore actions are captured and recorded in AWS CloudTrail.
- Backups
  - Each time you create an on-demand backup, the entire table data is backed up.
  - All backups and restores in DynamoDB work without consuming any provisioned throughput on the table.
  - DynamoDB backups do not guarantee causal consistency across items; however, the skew between updates in a backup is usually much less than a second.
  - You can restore backups as new DynamoDB tables in other regions.
- Restore
  - You cannot overwrite an existing table during a restore operation.
  - You restore backups to a new table.
  - For tables with even data distribution across your primary keys, the restore time is proportional to the largest single partition by item count and not the overall table size.
  - If your source table contains data with significant skew, the time to restore may increase.

Security

- Encryption
    - Encrypts your data at rest using an AWS Key Management Service (AWS KMS) managed encryption key for DynamoDB.
    - Encryption at rest can be enabled only when you are creating a new DynamoDB table.
    - After encryption at rest is enabled, it can't be disabled.
    - Uses AES-256 encryption.
    - Authentication and Access Control
        - Access to DynamoDB requires credentials.
        - Aside from valid credentials, you also need to have permissions to create or access DynamoDB resources.
        - Types of Identities
            - **AWS account root user**
            - **IAM user**
            - **IAM role**

Monitoring

- Automated tools:
    - **Amazon CloudWatch Alarms** – Watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods.
    - **Amazon CloudWatch Logs** – Monitor, store, and access your log files from AWS CloudTrail or other sources.
    - **Amazon CloudWatch Events** – Match events and route them to one or more target functions or streams to make changes, capture state information, and take corrective action.
    - **AWS CloudTrail Log Monitoring** – Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs, write log processing applications in Java, and validate that your log files have not changed after delivery by CloudTrail.
- Using the information collected by CloudTrail, you can determine the request that was made to DynamoDB, the IP address from which the request was made, who made the request, when it was made, and additional details.

Best Practices

- Know the Differences Between Relational Data Design and NoSQL

Relational database systems (RDBMS)	NoSQL database
In RDBMS, data can be queried flexibly, but queries are relatively expensive and don't scale well in high-traffic situations.	In a NoSQL database such as DynamoDB, data can be queried efficiently in a limited number of ways, outside of which queries can be expensive and slow.
In RDBMS, you design for flexibility without worrying about implementation details or performance. Query optimization generally doesn't affect schema design, but normalization is very important.	In DynamoDB, you design your schema specifically to make the most common and important queries as fast and as inexpensive as possible. Your data structures are tailored to the specific requirements of your business use cases.
For an RDBMS, you can go ahead and create a normalized data model without thinking about access patterns. You can then extend it later when new questions and query requirements arise. You can organize each type of data into its own table.	For DynamoDB, by contrast, you shouldn't start designing your schema until you know the questions it will need to answer. Understanding the business problems and the application use cases up front is essential. You should maintain as few tables as possible in a DynamoDB application. Most well designed applications require **only one** table.
	It is important to understand three fundamental properties of your application's access patterns: 1. Data size: Knowing how much data will be stored and requested at one time will help determine the most effective way to partition the data. 2. Data shape: Instead of reshaping data when a query is processed, a NoSQL database organizes data so that its shape in the database corresponds with what will be queried. 3. Data velocity: DynamoDB scales by increasing the number of physical partitions that are available to process queries, and by efficiently distributing data across those partitions. Knowing in advance what the peak query loads might be helps determine how to partition data to best use I/O capacity.

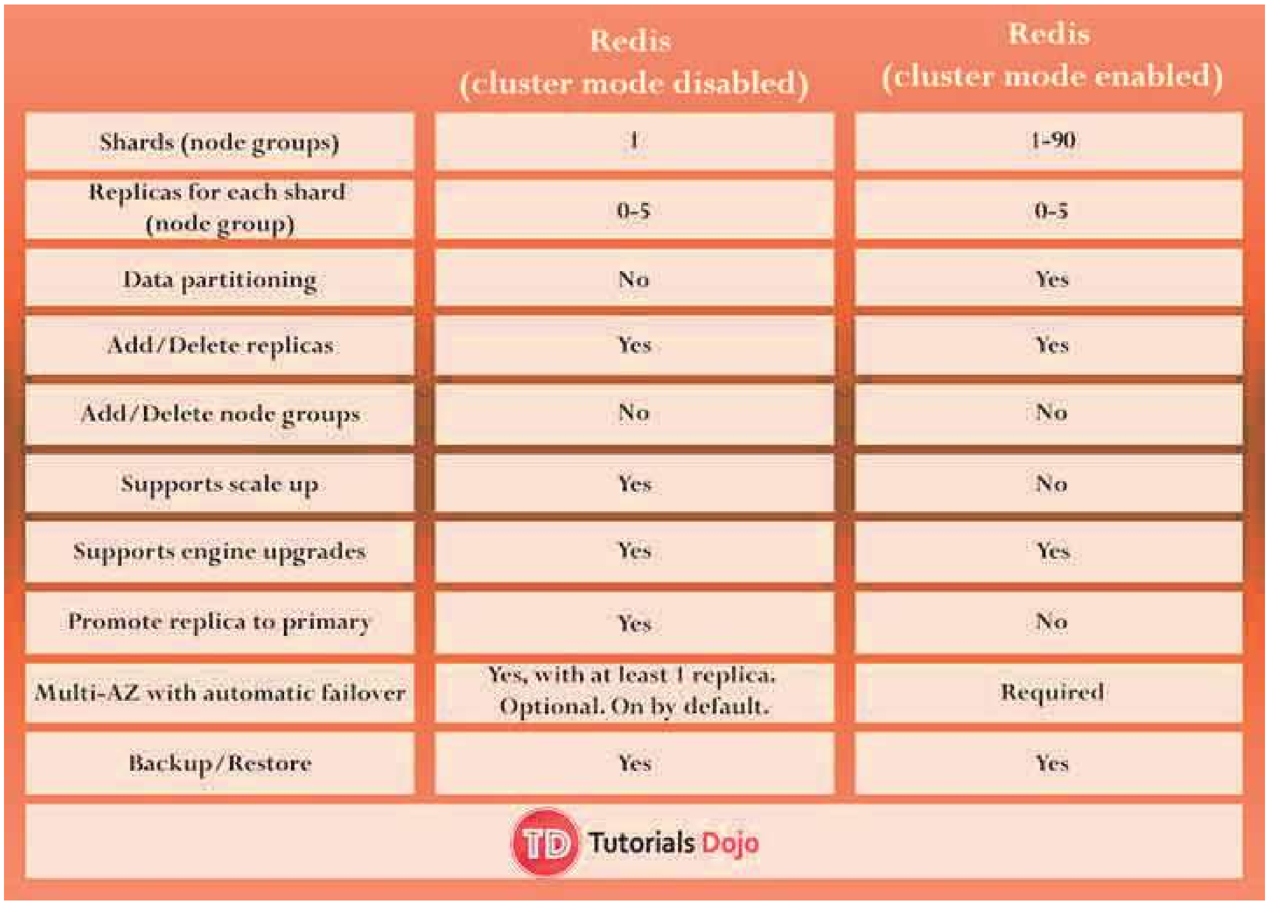
Pricing

- DynamoDB charges per GB of disk space that your table consumes. The first 25 GB consumed per month is free.
- DynamoDB charges for Provisioned Throughput ---- WCU and RCU, Reserved Capacity and Data Transfer Out.
- You should round up to the nearest KB when estimating how many capacity units to provision.
- There are additional charges for DAX, Global Tables, On-demand Backups (per GB), Continuous backups and point-in-time recovery (per GB), Table Restorations (per GB), and Streams (read request units).

Sources:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html?shortFooter=true
https://aws.amazon.com/dynamodb/faqs/

Amazon Elasticache

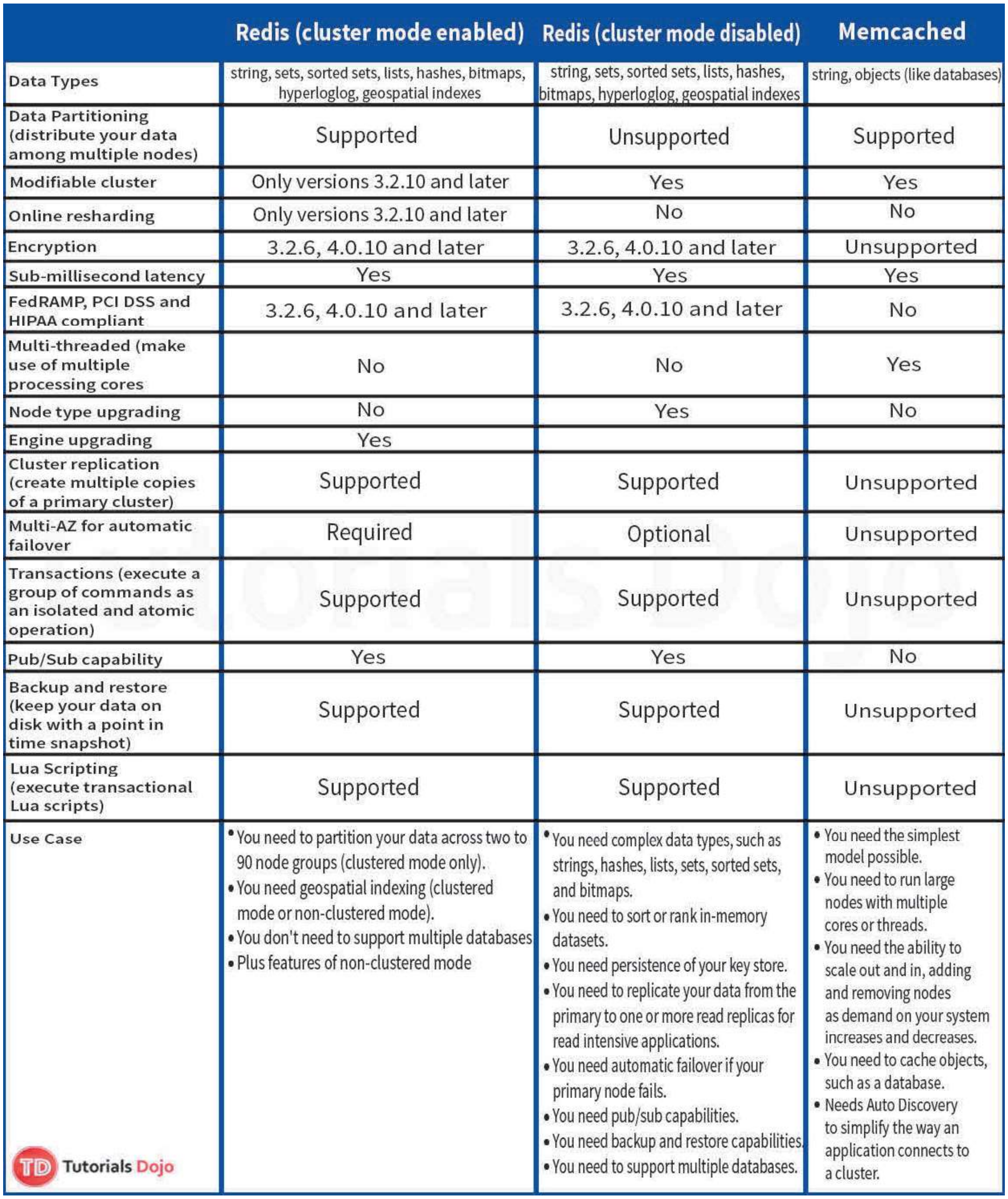
- ElastiCache is a distributed **in-memory cache** environment in the AWS Cloud.
- ElastiCache works with both the **Redis** and **Memcached** engines.
- Elasticache can be used for storing session state.



	Redis (cluster mode disabled)	Redis (cluster mode enabled)
Shards (node groups)	1	1-90
Replicas for each shard (node group)	0-5	0-5
Data partitioning	No	Yes
Add/Delete replicas	Yes	Yes
Add/Delete node groups	No	No
Supports scale up	Yes	No
Supports engine upgrades	Yes	Yes
Promote replica to primary	Yes	No
Multi-AZ with automatic failover	Yes, with at least 1 replica. Optional. On by default.	Required
Backup/Restore	Yes	Yes

Tutorials Dojo

- Redis VS Memcached
    - Memcached is designed for **simplicity** while Redis offers a **rich set of features** that make it effective for a wide range of use cases.



	Redis (cluster mode enabled)	Redis (cluster mode disabled)	Memcached
Data Types	string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog, geospatial indexes	string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog, geospatial indexes	string, objects (like databases)
Data Partitioning (distribute your data among multiple nodes)	Supported	Unsupported	Supported
Modifiable cluster	Only versions 3.2.10 and later	Yes	Yes
Online resharding	Only versions 3.2.10 and later	No	No
Encryption	3.2.6, 4.0.10 and later	3.2.6, 4.0.10 and later	Unsupported
Sub-millisecond latency	Yes	Yes	Yes
FedRAMP, PCI DSS and HIPAA compliant	3.2.6, 4.0.10 and later	3.2.6, 4.0.10 and later	No
Multi-threaded (make use of multiple processing cores	No	No	Yes
Node type upgrading	No	Yes	No
Engine upgrading	Yes		
Cluster replication (create multiple copies of a primary cluster)	Supported	Supported	Unsupported
Multi-AZ for automatic failover	Required	Optional	Unsupported
Transactions (execute a group of commands as an isolated and atomic operation)	Supported	Supported	Unsupported
Pub/Sub capability	Yes	Yes	No
Backup and restore (keep your data on disk with a point in time snapshot)	Supported	Supported	Unsupported
Lua Scripting (execute transactional Lua scripts)	Supported	Supported	Unsupported
Use Case	• You need to partition your data across two to 90 node groups (clustered mode only). • You need geospatial indexing (clustered mode or non-clustered mode). • You don't need to support multiple databases • Plus features of non-clustered mode	• You need complex data types, such as strings, hashes, lists, sets, sorted sets, and bitmaps. • You need to sort or rank in-memory datasets. • You need persistence of your key store. • You need to replicate your data from the primary to one or more read replicas for read intensive applications. • You need automatic failover if your primary node fails. • You need pub/sub capabilities. • You need backup and restore capabilities. • You need to support multiple databases.	• You need the simplest model possible. • You need to run large nodes with multiple cores or threads. • You need the ability to scale out and in, adding and removing nodes as demand on your system increases and decreases. • You need to cache objects, such as a database. • Needs Auto Discovery to simplify the way an application connects to a cluster.

- Pricing

- With on-demand nodes you pay only for the resources you consume by the hour without any long-term commitments.
- With Reserved Nodes, you can make a low, one-time, up-front payment for each node you wish to reserve for a 1 or 3 year term. In return, you receive a significant discount off the ongoing hourly usage rate for the Node(s) you reserve.
- ElastiCache provides storage space for one snapshot free of charge for each active ElastiCache for Redis cluster. Additional backup storage is charged.
- EC2 Regional Data Transfer charges apply when transferring data between an EC2 instance and an ElastiCache Node in different Availability Zones of the same Region.

Sources:
https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/
https://aws.amazon.com/elasticache/redis-details/
https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/
https://aws.amazon.com/elasticache/redis-vs-memcached/
https://aws.amazon.com/elasticache/features/
https://aws.amazon.com/elasticache/pricing/

Amazon Redshift

- A fully managed, **petabyte-scale data warehouse** service.
- Redshift extends data warehouse queries to your data lake. You can run analytic queries against petabytes of data stored locally in Redshift, and directly against exabytes of data stored in S3.
- RedShift is an OLAP type of DB.
- Currently, Redshift only supports Single-AZ deployments.
- Features
  - Redshift uses **columnar storage**, data compression, and zone maps to reduce the amount of I/O needed to perform queries.
  - It uses a **massively parallel processing** data warehouse architecture to parallelize and distribute SQL operations.
  - Redshift uses machine learning to deliver high throughput based on your workloads.
  - Redshift uses **result caching** to deliver sub-second response times for repeat queries.
  - Redshift automatically and continuously backs up your data to S3. It can asynchronously replicate your snapshots to S3 in another region for disaster recovery.
- Security
  - By default, an Amazon Redshift cluster is only accessible to the AWS account that creates the cluster.
  - Use IAM to create user accounts and manage permissions for those accounts to control cluster operations.
  - If you are using the EC2-Classic platform for your Redshift cluster, you must use Redshift security groups.
  - If you are using the EC2-VPC platform for your Redshift cluster, you must use VPC security groups.
  - When you provision the cluster, you can optionally choose to encrypt the cluster for additional security. Encryption is an immutable property of the cluster.
  - Snapshots created from the encrypted cluster are also encrypted.
- Pricing
  - You pay a per-second billing rate based on the type and number of nodes in your cluster.
  - You pay for the number of bytes scanned by RedShift Spectrum
  - You can reserve instances by committing to using Redshift for a 1 or 3 year term and save costs.

Sources:
https://docs.aws.amazon.com/redshift/latest/mgmt/
https://aws.amazon.com/redshift/features/
https://aws.amazon.com/redshift/pricing/
https://aws.amazon.com/redshift/faqs/