Database Services & Analytics

[Edition 02]

[Last Update 210509]

K21Academy
Learn Cloud from Experts

amazon web services

Contents

1 DOCUMENTATION LINK

1. Amazon DynamoDB

https://aws.amazon.com/dynamodb/

2. Amazon DynamoDB FAQs

https://aws.amazon.com/dynamodb/faqs/

3. Amazon DynamoDB Features

https://aws.amazon.com/dynamodb/features/

4. Amazon DynamoDB Pricing

https://aws.amazon.com/dynamodb/pricing/

5. Amazon Relational Database Service (RDS)

https://aws.amazon.com/rds/

6. Create and Connect to a MySQL Database

https://aws.amazon.com/getting-started/hands-on/create-mysql-db/

7. Creating an Amazon RDS DB Instance

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateDBInstance.html

8. Connecting to an Amazon RDS DB Instance

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_CommonTasks.Connect.html

9. Create a DB Instance

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Tutorials.WebServerDB.CreateDBInstance.html

2 AMAZON RDS

- Amazon Relational Database Service (Amazon RDS) is a managed service that makes it easy to set up, operate, and scale a relational database in the cloud.
- RDS is an Online Transaction Processing (OLTP) type of database.
- Primary use case is a transactional database (rather than analytical).
- Best for structured, relational data store requirements.
- Aims to be drop-in replacement for existing on-premise instances of the same databases.
- Automated backups and patching applied in customer-defined maintenance windows.
- Push-button scaling, replication and redundancy.

Amazon RDS supports the following database engines:

- Amazon Aurora
- MySQL
- MariaDB
- Oracle
- SQL Server
- PostgreSQL

- RDS is a managed service and you do not have access to the underlying EC2 instance (no root access).

The RDS service includes the following:

- Security and patching of the DB instances.
- Automated backup for the DB instances.
- Software updates for the DB engine.
- Easy scaling for storage and compute.
- Multi-AZ option with synchronous replication.
- Automatic failover for Multi-AZ option.
- Read replicas option for read heavy workloads.

- A DB instance is a database environment in the cloud with the compute and storage resources you specify.
- Database instances are accessed via endpoints.
- Endpoints can be retrieved via the DB instance description in the AWS Management Console, DescribeDBInstances API or describe-db-instances command.
- By default, customers are allowed to have up to a total of 40 Amazon RDS DB instances (only 10 of these can be Oracle or MS SQL unless you have your own licences).
- Maintenance windows are configured to allow DB instances modifications to take place such as scaling and software patching (some operations require the DB instance to be taken offline briefly).
- You can define the maintenance window or AWS will schedule a 30-minute window.
- Windows integrated authentication for SQL only works with domains created using the AWS directory service – need to establish a trust with an on-premise AD directory.

Events and Notifications:

- Amazon RDS uses AWS SNS to send RDS events via SNS notifications.
- You can use API calls to the Amazon RDS service to list the RDS events in the last 14 days (DescribeEvents API).
- You can view events from the last 14 days using the CLI.
- Using the AWS Console, you can only view RDS events for the last 1 day.

2.1 Use Cases, Alternatives & Anti-Patterns

The table below provides guidance on when best to use RDS and several other AWS database/data store services:

Data Store	When to Use
Database on EC2	• Ultimate control over database • Preferred DB not available under RDS
Amazon RDS	• Need traditional relational database for OLTP • Your data is well-formed and structured • Existing apps requiring RDBMS
Amazon DynamoDB	• Name/value pair data or unpredictable data structure • In-memory performance with persistence • High I/O needs • Scale dynamically
Amazon RedShift	• Massive amounts of data • Primarily OLAP workloads
Amazon Neptune	• Relationships between objects a major portion of data value
Amazon ElastiCache	• Fast temporary storage for small amounts of data • Highly volatile data
Amazon S3	• BLOBs • Static websites

Alternative to Amazon RDS:

If your use case isn't supported on RDS, you can run databases on Amazon EC2.

Consider the following points when considering a DB on EC2:

- You can run any database you like with full control and ultimate flexibility.
- You must manage everything like backups, redundancy, patching and scaling.

- Good option if you require a database not yet supported by RDS, such as IBM DB2 or SAP HANA.
- Good option if it is not feasible to migrate to AWS-managed database.

Anti-Patterns:

Anti-patterns are certain patterns in architecture or development that are considered bad, or sub-optimal practices – i.e. there may be a better service of method to produce the best result.

The following table describes requirements that are not a good fit for RDS:

Requirement	More Suitable Service
Lots of large binary objects (BLOBs)	S3
Automated Scalability	DynamoDB
Name/Value Data Structure	DynamoDB
Data is not well structured or unpredictable	DynamoDB
Other database platforms like IBM DB2 or SAP HANA	EC2
Complete control over the database	EC2

2.2 Encryption

- You can encrypt your Amazon RDS instances and snapshots at rest by enabling the encryption option for your Amazon RDS DB instance.
- Encryption at rest is supported for all DB types and uses AWS KMS.

When using encryption at rest the following elements are also encrypted:

- All DB snapshots
- Backups
- DB instance storage
- Read Replicas

- You cannot encrypt an existing DB, you need to create a snapshot, copy it, encrypt the copy, then build an encrypted DB from the snapshot.
- Data that is encrypted at rest includes the underlying storage for a DB instance, its automated backups, Read Replicas, and snapshots.
- A Read Replica of an Amazon RDS encrypted instance is also encrypted using the same key as the master instance when both are in the same region.

- If the master and Read Replica are in different regions, you encrypt using the encryption key for that region.
- You can't have an encrypted Read Replica of an unencrypted DB instance or an unencrypted Read Replica of an encrypted DB instance.
- Encryption/decryption is handled transparently.
- RDS supports SSL encryption between applications and RDS DB instances.
- RDS generates a certificate for the instance.

2.3 DB Subnet Groups

- A DB subnet group is a collection of subnets (typically private) that you create in a VPC and that you then designate for your DB instances.
- Each DB subnet group should have subnets in at least two Availability Zones in a given region.
- It is recommended to configure a subnet group with subnets in each AZ (even for standalone instances).
- During the creation of an RDS instance you can select the DB subnet group and the AZ within the group to place the RDS DB instance in.
- You cannot pick the IP within the subnet that is allocated.

2.4 Billing & Provisioning

- **AWS Charge for:**
  - DB instance hours (partial hours are charged as full hours).
  - Storage GB/month.
  - I/O requests/month – for magnetic storage.
  - Provisioned IOPS/month – for RDS provisioned IOPS SSD.
  - Egress data transfer.
  - Backup storage (DB backups and manual snapshots).
- Backup storage for the automated RDS backup is free of charge up to the provisioned EBS volume size.
- However, AWS replicate data across multiple AZs and so you are charged for the extra storage space on S3.

For multi-AZ you are charged for:

- Multi-AZ DB hours.
- Provisioned storage.
- Double write I/Os.

- For multi-AZ you are not charged for DB data transfer during replication from primary to standby.
- Oracle and Microsoft SQL licences are included or you can bring your own (BYO).
- On-demand and reserved instance pricing available.
- Reserved instances are defined based on the following attributes which must not be changed:
  - DB engine.
  - DB instance class
  - Deployment type (standalone, multi-AZ_
  - License model
  - Region

Reserved instances:

- Can be moved between AZs in the same region.
- Are available for multi-AZ deployments.
- Can be applied to Read Replicas if DB instance class and region are the same.
- Scaling is achieved through changing the instance class for compute and modifying storage capacity for additional storage allocation.

2.5 Scalability

- You can only scale RDS up (compute and storage).
- You cannot decrease the allocated storage for an RDS instance.
- You can scale storage and change the storage type for all DB engines except MS SQL.
- For MS SQL the workaround is to create a new instance from a snapshot with the new configuration.
- Scaling storage can happen while the RDS instance is running without outage however there may be performance degradation.
- Scaling compute will cause downtime.
- You can choose to have changes take effect immediately, however the default is within the maintenance window.

- Scaling requests are applied during the specified maintenance window unless “apply immediately” is used.
- All RDS DB types support a maximum DB size of 64 TiB except for Microsoft SQL Server (16 TiB).

2.6 Performance

- Amazon RDS uses EBS volumes (never uses instance store) for DB and log storage.
- There are three storage types available: General Purpose (SSD), Provisioned IOPS (SSD), and Magnetic.

General Purpose (SSD):

- Use for Database workloads with moderate I/O requirement
- Cost effective
- Also called gp2
- 3 IOPS/GB
- Burst up to 3000 IOPS

Provisioned IOPS (SSD):

- Use for I/O intensive workloads
- Low latency and consistent I/O
- User specified IOPS (see table below)

For provisioned IOPS storage the table below shows the range of Provisioned IOPS and storage size range for each database engine.

Database Engine	Range of Provisioned IOPS	Range of Storage
MariaDB	1,000–40,000 IOPS	100 GiB - 16 TiB
SQL Server, Enterprise and Standard editions	1,000–32,000 IOPS	200 GiB - 16 TiB
SQL Server, Web and Express editions	1,000–32,000 IOPS	100 GiB - 16 TiB
MySQL	1,000–40,000 IOPS	100 GiB - 16 TiB
Oracle	1,000–40,000 IOPS	100 GiB - 16 TiB
PostgreSQL	1,000–40,000 IOPS	100 GiB - 16 TiB

Magnetic:

- Not recommended anymore, available for backwards compatibility.
- 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 4 TiB.
- Limited to a maximum of 1,000 IOPS.

2.7 Multi AZ & Read Replicas

- Multi-AZ and Read Replicas are used for high availability, fault tolerance and performance scaling.

The table below compares multi-AZ deployments to Read Replicas:

Multi-AZ Deployments	Read Replicas
Synchronous replication – highly durable	Asynchronous replication – highly scalable
Only database engine on primary instance is active	All read replicas are accessible 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

2.8 Multi-AZ

- Multi-AZ RDS creates a replica in another AZ and synchronously replicates to it (DR only).
- There is an option to choose multi-AZ during the launch wizard.
- AWS recommends the use of provisioned IOPS storage for multi-AZ RDS DB instances.
- Each AZ runs on its own physically distinct, independent infrastructure, and is engineered to be highly reliable.
- You cannot choose which AZ in the region will be chosen to create the standby DB instance.

- You can view which AZ the standby DB instance is created in.
- A failover may be triggered in the following circumstances:
  - Loss of primary AZ or primary DB instance failure.
  - Loss of network connectivity on primary.
  - Compute (EC2) unit failure on primary.
  - Storage (EBS) unit failure on primary.
  - The primary DB instance is changed.
  - Patching of the OS on the primary DB instance.
  - Manual failover (reboot with failover selected on primary).
- During failover RDS automatically updates configuration (including DNS endpoint) to use the second node.
- Depending on the instance class it can take 1 to a few minutes to failover to a standby DB instance.
- It is recommended to implement DB connection retries in your application.
- Recommended to use the endpoint rather than the IP address to point applications to the RDS DB.
- The method to initiate a manual RDS DB instance failover is to reboot selecting the option to failover.
- A DB instance reboot is required for changes to take effect when you change the DB parameter group or when you change a static DB parameter.
- The DB parameter group is a configuration container for the DB engine configuration.
- You will be alerted by a DB instance event when a failover occurs.
- There is no charge for data transfer between primary and secondary RDS instances.
- Multi-AZ deployments for the MySQL, MariaDB, Oracle and PostgreSQL engines use Amazon's failover technology.
- Multi-AZ deployments for the SQL Server engine use SQL Server Database Mirroring (DBM).
- System upgrades like OS patching, DB Instance scaling and system upgrades, are applied first on the standby, before failing over and modifying the other DB Instance.
- In multi-AZ configurations snapshots and automated backups are performed on the standby to avoid I/O suspension on the primary instance.

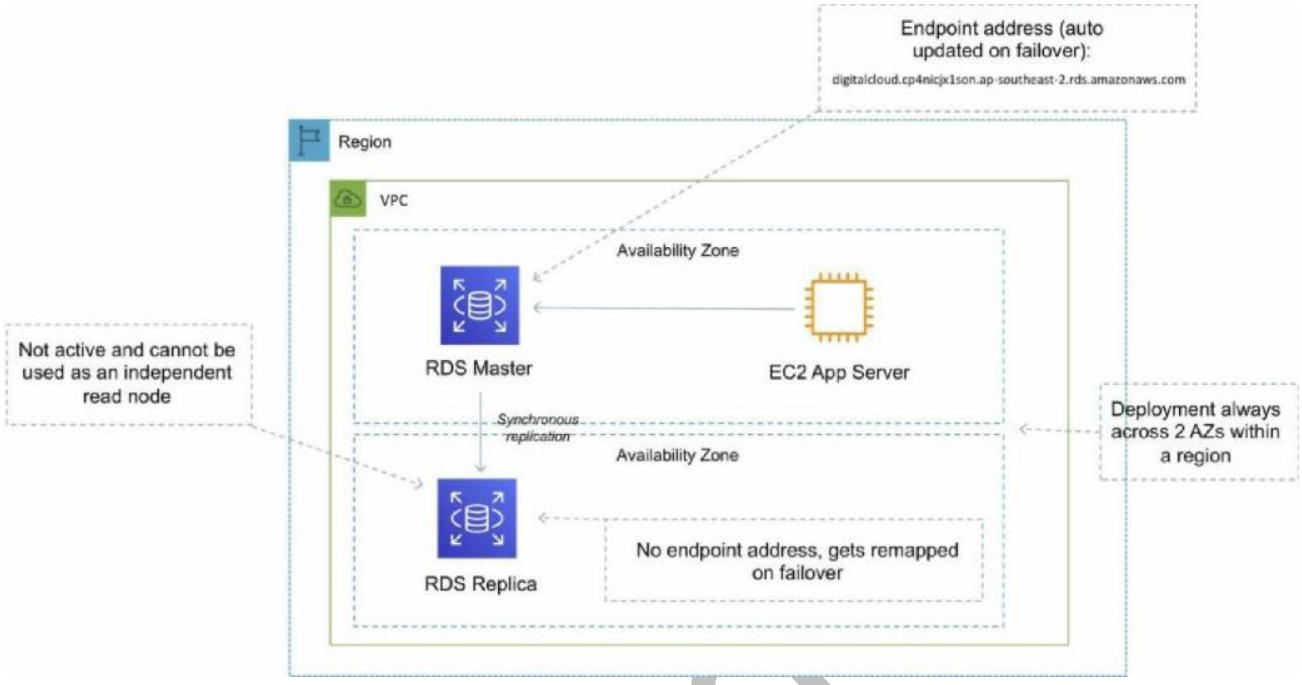
Read Replica Support for Multi-AZ:

- Amazon RDS Read Replicas for MySQL and MariaDB support Multi-AZ deployments.
- Combining Read Replicas with Multi-AZ enables you to build a resilient disaster recovery strategy and simplify your database engine upgrade process.
- A Read Replica in a different region than the source database can be used as a standby database and promoted to become the new production database in case of a regional disruption.
- This allows you to scale reads whilst also having multi-AZ for DR.
- Note that RDS for PostgreSQL does not yet support this feature.

The process for implementing maintenance activities is as follows:

- Perform operations on standby.
- Promote standby to primary.
- Perform operations on new standby (demoted primary).

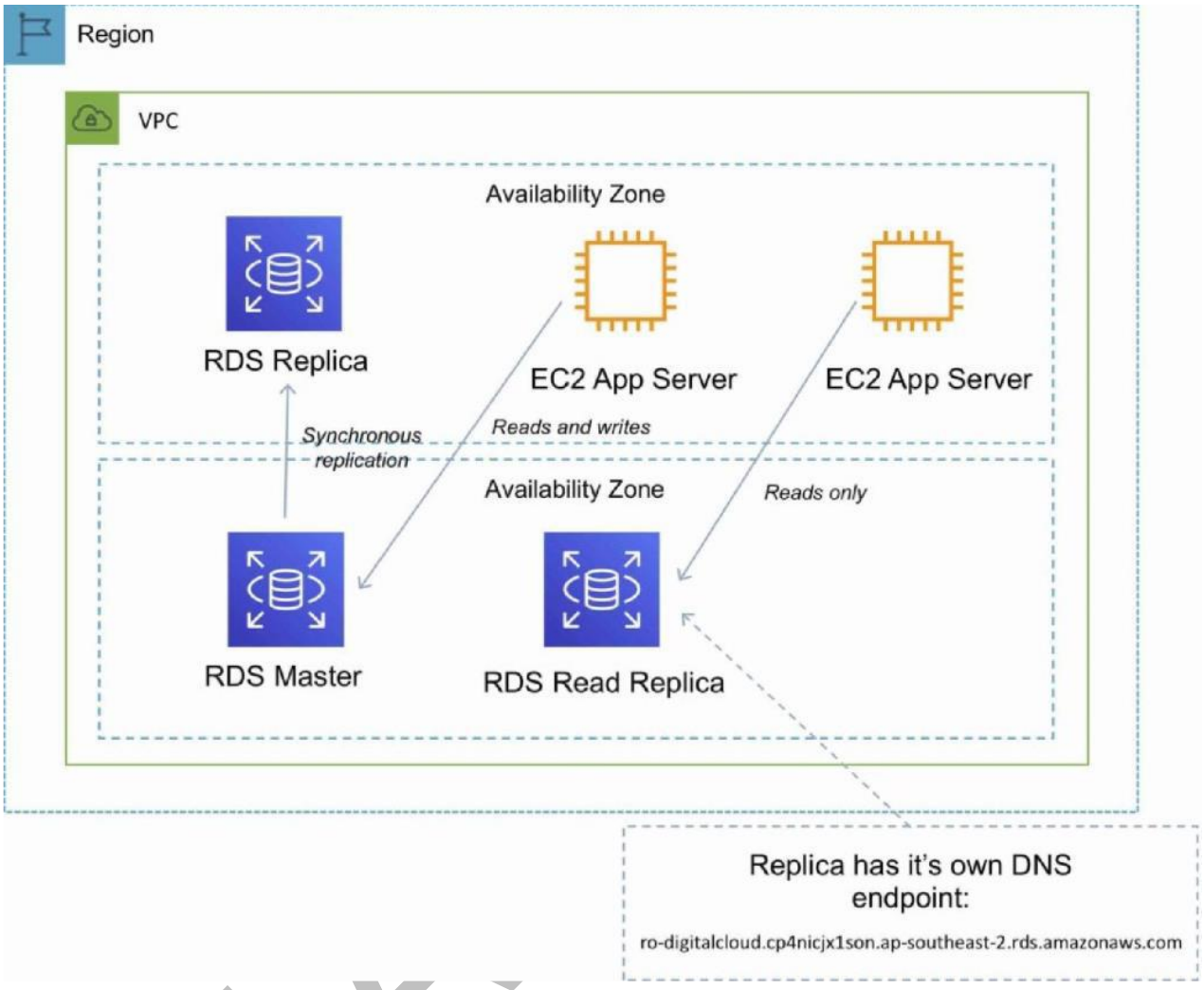
- You can manually upgrade a DB instance to a supported DB engine version from the AWS Console.
- By default, upgrades will take effect during the next maintenance window.
- You can optionally force an immediate upgrade.
- In multi-AZ deployments version upgrades will be conducted on both the primary and standby at the same time causing an outage of both DB instance.
- Ensure security groups and NACLs will allow your application servers to communicate with both the primary and standby instances.



2.9 Read Replicas

- Read replicas are used for read-heavy DBs and replication is asynchronous.
- Read replicas are for workload sharing and offloading.
- Read replicas provide read-only DR.
- Read replicas are created from a snapshot of the master instance.
- Must have automated backups enabled on the primary (retention period > 0).
- Only supported for transactional database storage engines (InnoDB not MyISAM).
- Read replicas are available for MySQL, PostgreSQL, MariaDB, Oracle and Aurora (not SQL Server).
- For the MySQL, MariaDB, PostgreSQL, and Oracle database engines, Amazon RDS creates a second DB instance using a snapshot of the source DB instance.
- It then uses the engines' native asynchronous replication to update the read replica whenever there is a change to the source DB instance.
- Amazon Aurora employs an SSD-backed virtualized storage layer purpose-built for database workloads.
- You can take snapshots of PostgreSQL read replicas but cannot enable automated backups.
- You can enable automatic backups on MySQL and MariaDB read replicas.
- You can enable writes to the MySQL and MariaDB Read Replicas.

- You can have 5 read replicas of a production DB.
- You cannot have more than four instances involved in a replication chain.
- You can have read replicas of read replicas for MySQL and MariaDB but not for PostgreSQL.
- Read replicas can be configured from the AWS Console or the API.
- You can specify the AZ the read replica is deployed in.
- The read replicas storage type and instance class can be different from the source but the compute should be at least the performance of the source.
- You cannot change the DB engine.
- In a multi-AZ failover, the read replicas are switched to the new primary.
- Read replicas must be explicitly deleted.
- If a source DB instance is deleted without deleting the replicas each replica becomes a standalone single-AZ DB instance.
- You can promote a read replica to primary.
- Promotion of read replicas takes several minutes.
- **Promoted read replicas retain:**
  - Backup retention window.
  - Backup window.
  - DB parameter group.
- Existing read replicas continue to function as normal.
- Each read replica has its own DNS endpoint.
- Read replicas can have multi-AZ enabled and you can create read replicas of multi-AZ source DBs.
- Read replicas can be in another region (uses asynchronous replication).
- This configuration can be used for centralizing data from across different regions for analytics.



2.10 DB Snapshots

- DB Snapshots are user-initiated and enable you to back up your DB instance in a known state as frequently as you wish, and then restore to that specific state.
- Cannot be used for point-in-time recovery.
- Snapshots are stored on S3.
- Snapshots remain on S3 until manually deleted.
- Backups are taken within a defined window.
- I/O is briefly suspended while backups initialize and may increase latency (applicable to single-AZ RDS).
- DB snapshots that are performed manually will be stored even after the RDS instance is deleted.

- Restored DBs will always be a new RDS instance with a new DNS endpoint.
- Can restore up to the last 5 minutes.
- You cannot restore from a DB snapshot to an existing DB – a new instance is created when you restore.
- Only default DB parameters and security groups are restored – you must manually associate all other DB parameters and SGs.
- It is recommended to take a final snapshot before deleting an RDS instance.
- Snapshots can be shared with other AWS accounts.

2.11 High Availability Approaches For Databases

- If possible, choose DynamoDB over RDS because of inherent fault tolerance.
- If DynamoDB can't be used, choose Aurora because of redundancy and automatic recovery features.
- If Aurora can't be used, choose Multi-AZ RDS.
- Frequent RDS snapshots can protect against data corruption or failure and they won't impact performance of Multi-AZ deployment.
- Regional replication is also an option but will not be strongly consistent.
- If the database runs on EC2, you have to design the HA yourself.

2.12 Migration

- AWS Database Migration Service helps you migrate databases to AWS quickly and securely.
- Use along with the Schema Conversion Tool (SCT) to migrate databases to AWS RDS or EC2-based databases.
- The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database.
- The AWS Database Migration Service can migrate your data to and from most widely used commercial and open-source databases.
- Schema Conversion Tool can copy database schemas for homogenous migrations (same database) and convert schemas for heterogeneous migrations (different database).
- DMS is used for smaller, simpler conversions and also supports MongoDB and DynamoDB.
- SCT is used for larger, more complex datasets like data warehouses.
- DMS has replication functions for on-premise to AWS or to Snowball or S3.

2.13 Sample Questions

Q1: You have been tasked with ensuring that data stored in your organization's RDS instance exists in a minimum of two geographically distributed locations. Which of the following solutions are valid approaches? (Choose two.)

A. Enable RDS in a Multi-AZ configuration.

B. Enable RDS in a read replica configuration.

C. Install a storage gateway with stored volumes.

D. Enable RDS in a cross-region read replica configuration.

Answer: A, D

Explanation: A Multi-AZ setup is the easiest solution, and the most common. Turning on read replicas (option B) is not a guarantee, as read replicas are not automatically installed in different AZs or regions. However, with option D, a cross-region replica configuration will ensure multiple regions are used. A storage gateway (option C) is backed by S3, not RDS.

Q2 : What type of services are associated with RDS?

A. Storage services

B. Database services

C. Compute services

D. Networking services

Answer: B

Explanation: RDS is the Relational Database Service, which provides managed database services for your applications.

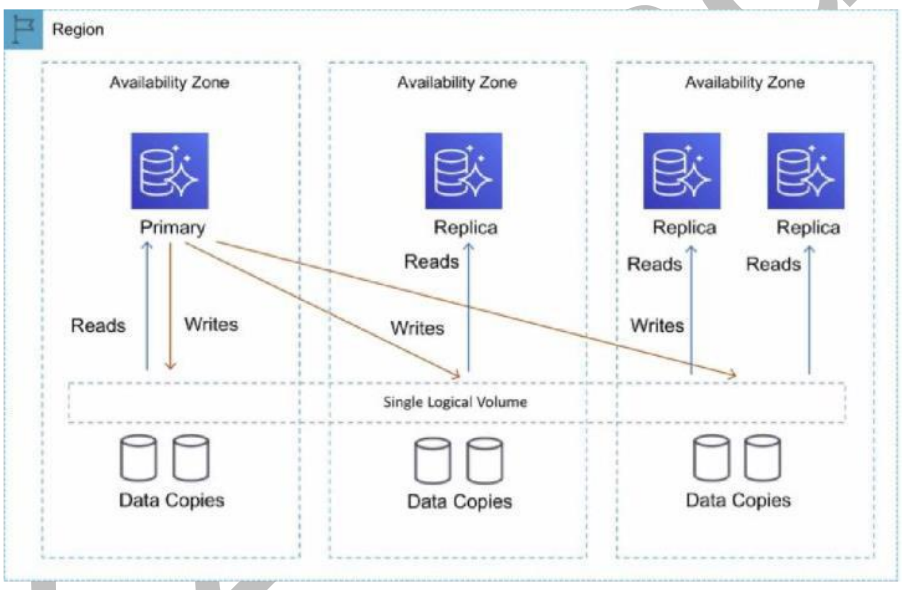
For more Questions Please check Certification Sample Quiz under each module

Link: https://k21academy.com/awssaquizm08

3 AMAZON AURORA

- Amazon Aurora is a relational database service that combines the speed and availability of high-end commercial databases with the simplicity and cost-effectiveness of open source databases.
- Aurora is an AWS proprietary database.
- Fully managed service.
- High performance, low price.
- Scales in 10GB increments.
- Scales up to 32vCPUs and 244GB RAM.
- 2 copies of data are kept in each AZ with a minimum of 3 AZ's (6 copies).
- Can handle the loss of up to two copies of data without affecting DB write availability and up to three copies without affecting read availability.

The following diagram depicts how Aurora Fault Tolerance and Replicas work:



Aurora Fault Tolerance
- Fault tolerance across 3 AZs
- Single logical volume
- Aurora Replicas scale-out read requests
- Up to 15 Aurora Replicas with sub-10ms replica lag
- Aurora Replicas are independent endpoints
- Can promote Aurora Replica to be a new primary or create new primary
- Set priority (tiers) on Aurora Replicas to control order of promotion
- Can use Auto Scaling to add replicas

3.1 Aurora Replicas

- There are two types of replication: Aurora replica (up to 15), MySQL Read Replica (up to 5).
- The table below describes the differences between the two replica options:

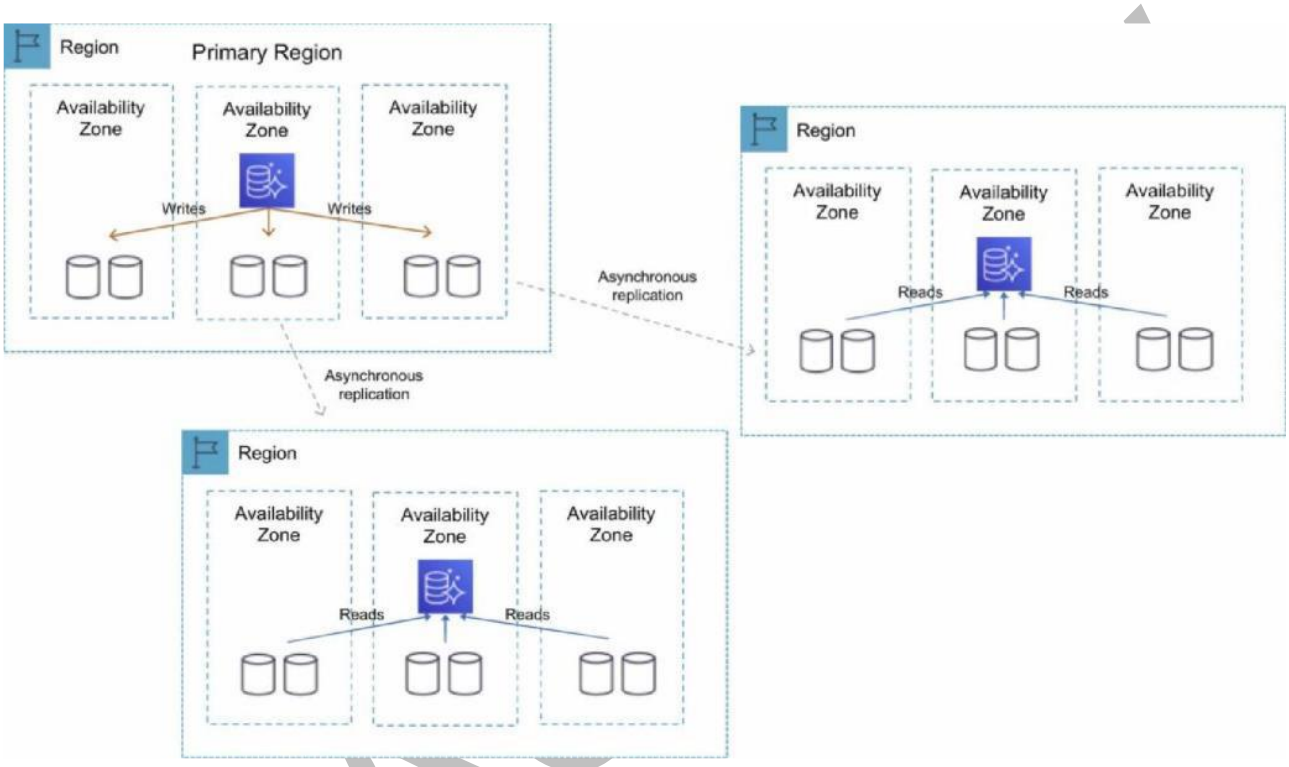
Feature	Aurora Replica	MySQL Replica
Number of replicas	Up to 15	Up to 5
Replication type	Asynchronous (milliseconds)	Asynchronous (seconds)
Performance impact on primary	Low	High
Replica location	In-region	Cross-region
Act as failover target	Yes (no data loss)	Yes (potentially minutes of data loss)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

- You can create read replicas for an Amazon Aurora database in up to five AWS regions. This capability is available for Amazon Aurora with MySQL compatibility.

3.2 Cross-Region Read Replicas

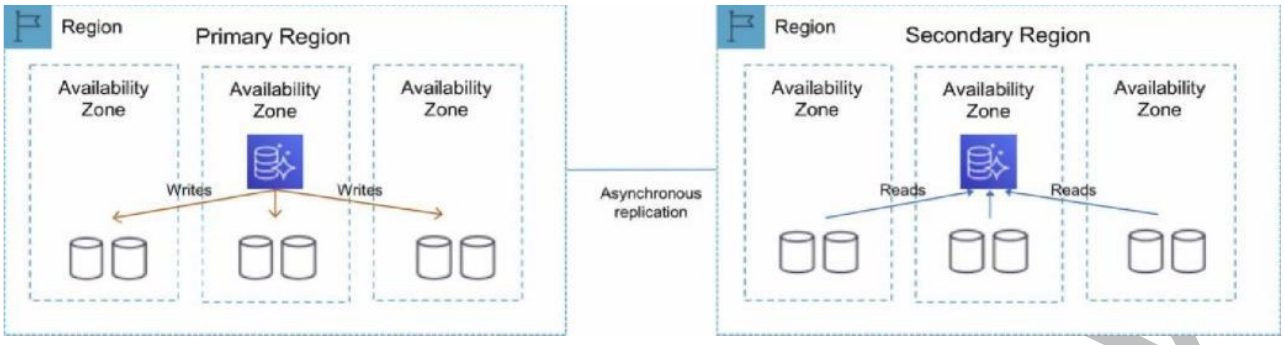
- Cross-region read replicas allow you to improve your disaster recovery posture, scale read operations in regions closer to your application users, and easily migrate from one region to another.
- Cross-region replicas provide fast local reads to your users.
- Each region can have an additional 15 Aurora replicas to further scale local reads.

- You can choose between Global Database, which provides the best replication performance, and traditional binlog-based replication.
- You can also set up your own binlog replication with external MySQL databases.
- The following diagram depicts the Cross-Region Read Replica topology:



3.3 Global Database

- For globally distributed applications you can use Global Database, where a single Aurora database can span multiple AWS regions to enable fast local reads and quick disaster recovery.
- Global Database uses storage-based replication to replicate a database across multiple AWS Regions, with typical latency of less than 1 second.
- You can use a secondary region as a backup option in case you need to recover quickly from a regional degradation or outage.
- A database in a secondary region can be promoted to full read/write capabilities in less than 1 minute.
- **The following table depicts the Aurora Global Database topology:**



3.4 Multi Master

- Amazon Aurora Multi-Master is a new feature of the Aurora MySQL-compatible edition that adds the ability to scale out write performance across multiple Availability Zones, allowing applications to direct read/write workloads to multiple instances in a database cluster and operate with higher availability.
- Aurora Multi-Master is designed to achieve high availability and ACID transactions across a cluster of database nodes with configurable read after write consistency.
- **Architecture:**
  - An Aurora cluster consists of a set of compute (database) nodes and a shared storage volume.
  - The storage volume consists of six storage nodes placed in three Availability Zones for high availability and durability of user data.
  - Every database node in the cluster is a writer node that can run read and write statements.
- There is no single point of failure in the cluster.
- Applications can use any writer node for their read/write and DDL needs.
- A database change made by a writer node is written to six storage nodes in three Availability Zones, providing data durability and resiliency against storage node and Availability Zone failures.
- The writer nodes are all functionally equal, and a failure of one writer node does not affect the availability of the other writer nodes in the cluster.
- **High Availability:**
  - Aurora Multi-Master improves upon the high availability of the single-master version of Amazon Aurora because all of the nodes in the cluster are read/write nodes.
  - With single-master Aurora, a failure of the single writer node requires the promotion of a read replica to be the new writer.

- In the case of Aurora Multi-Master, the failure of a writer node merely requires the application using the writer to open connections to another writer.

3.5 Aurora Serverless

- Amazon Aurora Serverless is an on-demand, auto-scaling configuration for Amazon Aurora.
- Available for MySQL-compatible and PostgreSQL-compatible editions.
- The database automatically starts up, shuts down, and scales capacity up or down based on application needs.
- It enables you to run a database in the cloud without managing any database instances. It's a simple, cost-effective option for infrequent, intermittent, or unpredictable workloads.
- You simply create a database endpoint and optionally specify the desired database capacity range and connect applications.
- With Aurora Serverless, you only pay for database storage and the database capacity and I/O your database consumes while it is active.
- Pay on a per-second basis for the database capacity you use when the database is active.
- Can migrate between standard and serverless configurations with a few clicks in the Amazon RDS Management Console.
- The table below provides a few example use cases for Amazon Aurora Serverless:

Use Case	Example
Infrequently-Used Applications	Application that is only used for a few minutes several times per day or week. Need a cost-effective database that only requires you to pay when it's active. With Aurora Serverless, you only pay for the database resources you consume.
New Applications	Deploying a new application and are unsure which instance size you need. With Aurora Serverless, you simply create an end-point and let the database auto-scale to the capacity requirements of your application.
Variable Workloads	Running a lightly-used application, with peaks of 30 minutes to several hours a few times each day or several times per year. Now you only pay for what the resources needed based on load – avoiding paying for unused resources or risking poor performance.
Unpredictable Workloads	Running workloads where there is database usage throughout the day, and also peaks of activity that are hard to predict. With Aurora Serverless, your database will auto-scale capacity to meet the needs of the application's peak load and scale back down when the surge of activity is over.
Development and Test Databases	Software development and QA teams are using databases during work hours, but don't need them on nights or weekends. With Aurora Serverless, your database automatically shuts down when not in use, and starts up much more quickly when work starts the next day.
Multitenant Applications	Web-based application with a database for each of your customers. Now you don't have to manage database capacity individually for each application in your fleet. Aurora manages individual database capacity for you, saving you valuable time.

3.6 Fault-Tolerant & Self-Healing Storage

- Each 10GB chunk of your database volume is replicated six ways, across three Availability Zones.
- Amazon Aurora storage is fault-tolerant, transparently handling the loss of up to two copies of data without affecting database write availability and up to three copies without affecting read availability.
- Amazon Aurora storage is also self-healing; data blocks and disks are continuously scanned for errors and replaced automatically.

3.7 Aurora Auto Scaling

- Aurora Auto Scaling dynamically adjusts the number of Aurora Replicas provisioned for an Aurora DB cluster using single-master replication.
- Aurora Auto Scaling is available for both Aurora MySQL and Aurora PostgreSQL.
- Aurora Auto Scaling enables your Aurora DB cluster to handle sudden increases in connectivity or workload.
- When the connectivity or workload decreases, Aurora Auto Scaling removes unnecessary Aurora Replicas so that you don't pay for unused provisioned DB instances.

3.8 Automatic, Continuous, Incremental Backups & Point-In-Time Restore

- Amazon Aurora's backup capability enables point-in-time recovery for your instance.
- This allows you to restore your database to any second during your retention period, up to the last five minutes.
- Your automatic backup retention period can be configured up to thirty-five days.
- Automated backups are stored in Amazon S3, which is designed for 99.999999999% durability. Amazon Aurora backups are automatic, incremental, and continuous and have no impact on database performance.
- When automated backups are turned on for your DB Instance, Amazon RDS automatically performs a full daily snapshot of your data (during your preferred backup window) and captures transaction logs (as updates to your DB Instance are made).
- Automated backups are enabled by default and data is stored on S3 and is equal to the size of the DB.
- Amazon RDS retains backups of a DB Instance for a limited, user-specified period of time called the retention period, which by default is 7 days but can be up to 35 days.
- There are two methods to backup and restore RDS DB instances:
  - Amazon RDS automated backups.
  - User initiated manual backups.
- Both options back up the entire DB instance and not just the individual DBs.
- Both options create a storage volume snapshot of the entire DB instance.
- You can make copies of automated backups and manual snapshots.
- Automated backups backup data to multiple AZs to provide for data durability.

- Multi-AZ backups are taken from the standby instance (for MariaDB, MySQL, Oracle and PostgresSQL).
- The DB instance must be in an Active state for automated backups to happen.
- Only automated backups can be used for point-in-time DB instance recovery.
- The granularity of point-in-time recovery is 5 minutes.
- Amazon RDS creates a daily full storage volume snapshot and also captures transaction logs regularly.
- You can choose the backup window.
- There is no additional charge for backups but you will pay for storage costs on S3.
- You can disable automated backups by setting the retention period to zero (0).
- An outage occurs if you change the backup retention period from zero to a non-zero value or the other way around.
- The retention period is the period AWS keeps the automated backups before deleting them.
- Retention periods:
  - By default the retention period is 7 days if configured from the console for all DB engines except Aurora.
  - The default retention period is 1 day if configured from the API or CLI.
  - The retention period for Aurora is 1 day regardless of how it is configured.
  - You can increase the retention period up to 35 days.
- During the backup window I/O may be suspended.
- Automated backups are deleted when you delete the RDS DB instance.
- Automated backups are only supported for InnoDB storage engine for MySQL (not for myISAM).
- When you restore a DB instance the default DB parameters and security groups are applied – you must then apply the custom DB parameters and security groups.
- You cannot restore from a DB snapshot into an existing DB instance.
- Following a restore the new DB instance will have a new endpoint.
- The storage type can be changed when restoring a snapshot.

3.9 Sample Questions

Q1: In an RDS, managed service capacity, which of the following databases is generally fastest?

A. PostgreSQL

B. MySQL

C. Aurora

D. They are all equivalent.

Answer: C

Explanation: Aurora, under the RDS managed service, is about five times as fast as MySQL and three times as fast as PostgreSQL. Still, there's an easier way to remember this: Anytime an AWS exam asks you about speed or performance, it's generally the case that the AWS offering is the right answer. AWS won't ask you to choose MySQL or Oracle as a faster option than one of its own databases!

Q2 : In an RDS, managed service capacity, which of the following databases is most resistant to disaster by default?

A. Aurora

B. Oracle

C. MySQL

D. They are all equivalent.

Answer: A

Explanation: Aurora, under the RDS managed service, stores six copies of your data by default, across three availability zones. Additionally, there's an easier way to remember this: Anytime an AWS exam asks you about resilience, it's generally the case that the AWS offering is the right answer.

For more Questions Please check Certification Sample Quiz under each module

Link: https://k21academy.com/awssaquizm08

4 AMAZON DYNAMODB

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
- Multi-AZ NoSQL data store with Cross-Region Replication option.
- Push button scaling means that you can scale the DB at any time without incurring downtime.
- Defaults to eventual consistency reads but can request strongly consistent read via SDK parameter.
- Priced on throughput, rather than compute.
- Provision read and write capacity in anticipation of need.
- Autoscale capacity adjusts per configured min/max levels.
- On-Demand Capacity provides flexible capacity at a small premium cost.
- Can achieve ACID compliance with DynamoDB Transactions.
- SSD based and uses limited indexing on attributes for performance.
- DynamoDB is a Web service that uses HTTP over SSL (HTTPS) as a transport and JSON as a message serialization format.
- Amazon DynamoDB stores three geographically distributed replicas of each table to enable high availability and data durability.
- Data is synchronously replicated across 3 facilities (AZs) in a region.
- **Cross-region replication allows you to replicate across regions:**
  - Amazon DynamoDB global tables provides a fully managed solution for deploying a multi-region, multi-master database.
  - When you create a global table, you specify the AWS regions where you want the table to be available.
  - DynamoDB performs all of the necessary tasks to create identical tables in these regions, and propagate ongoing data changes to all of them.
- Provides low read and write latency.
- Scale storage and throughput up or down as needed without code changes or downtime.
- DynamoDB is schema-less.
- DynamoDB can be used for storing session state.
- Provides two read models.
- **Eventually consistent reads (Default):**

- The eventual consistency option maximises your read throughput (best read performance).
- An eventually consistent read might not reflect the results of a recently completed write.
- Consistency across all copies reached within 1 second.

- **Strongly consistent reads:**
  - A strongly consistent read returns a result that reflects all writes that received a successful response prior to the read (faster consistency).
- Users/applications reading from DynamoDB tables can specify in their requests if they want strong consistency (default is eventually consistent).
- Attributes consists of a name and a value or set of values.
- Attributes in DynamoDB are similar to fields or columns in other database systems.
- The primary key is the only required attribute for items in a table and it uniquely identifies each item.
- A primary key can either be one of the following types.

Partition key:

- A simple primary key composed of one attribute known as the partition key.

Partition key and sort key:

- Referred to as a composite primary key.
- Composed of two attributes: partition key and sort key.

- An item is a collection of attributes.
- The aggregate size of an item cannot exceed 400KB including keys and all attributes.
- Can store pointers to objects in S3, including items over 400KB.
- Tables are a collection of items and items are made up of attributes (columns).
- Supports key-value and document data structures.
- Supports fast, in-place Atomic updates.
- Stores structured data in tables, indexed by a primary key.
- Supports GET/PUT operations using a user-defined primary key.
- DynamoDB provides flexible querying by letting you query on non-primary key attributes using Global Secondary Indexes and Local Secondary Indexes.
- You can create one or more secondary indexes on a table.
- A secondary index lets you query the data in the table using an alternate key, in addition to queries against the primary key.
- **DynamoDB supports two kinds of secondary indexes:**

- Global secondary index – An index with a partition key and sort key that can be different from those on the table.
- Local secondary index – An index that has the same partition key as the table, but a different sort key.

- **You can search using one of the following methods:**
  - Query operation – find items in a table or a secondary index using only the primary keys attributes.
  - Scan operation – reads every item in a table or a secondary index and by default will return all items.
- Use DynamoDB when relational features are not required and the DB is likely to need to scale.
- Not ideal for the following situations:
  - Traditional RDS apps.
  - Joins and/or complex transactions.
  - BLOB data.
  - Large data with low I/O rate.

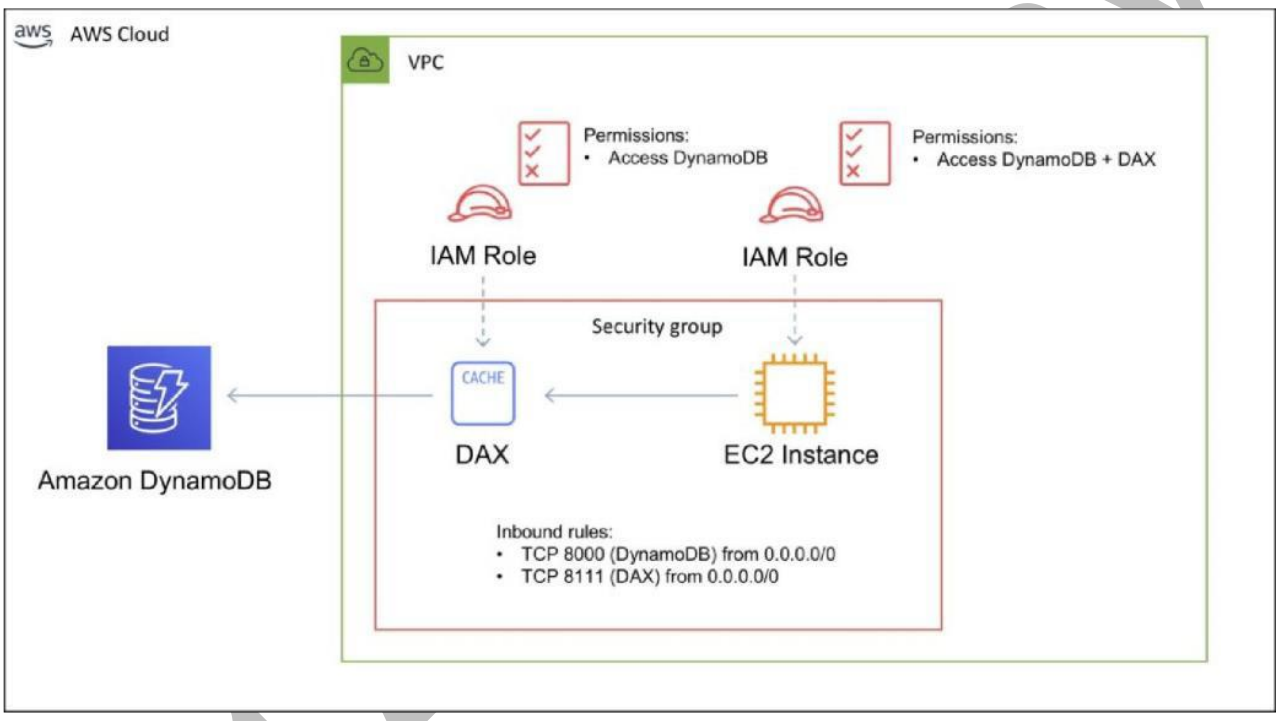
4.1 DynamoDB Streams

- DynamoDB Streams help you to keep a list of item level changes or provide a list of item level changes that have taken place in the last 24hrs.
- Amazon DynamoDB is integrated with AWS Lambda so that you can create triggers—pieces of code that automatically respond to events in DynamoDB Streams.
- If you enable DynamoDB Streams on a table, you can associate the stream ARN with a Lambda function that you write.

4.2 Dynamo DAX

- Amazon DynamoDB Accelerator (DAX) is a fully managed, highly available, in-memory cache for DynamoDB that delivers up to a 10x performance improvement.
- Improves performance from milliseconds to microseconds, even at millions of requests per second.
- DAX does all the heavy lifting required to add in-memory acceleration to your DynamoDB tables, without requiring developers to manage cache invalidation, data population, or cluster management.
- You do not need to modify application logic, since DAX is compatible with existing DynamoDB API calls.

- You can enable DAX with just a few clicks in the AWS Management Console or using the AWS SDK.
- Just as with DynamoDB, you only pay for the capacity you provision.
- Provisioned through clusters and charged by the node (runs on EC2 instances).
- Pricing is per node-hour consumed and is dependent on the instance type you select.
- The following diagram depicts the Amazon DynamoDB DAX service.



Note the following:

- ➢ You can apply an IAM role to the the DAX nodes
- ➢ You can apply Security Groups to the DAX nodes
- ➢ DynamoDB DAX sits within your VPC

4.3 Best Practices

- Keep item sizes small.
- If you are storing serial data in DynamoDB that will require actions based on date/time use separate tables for days, weeks, months.
- Store more frequently and less frequently accessed data in separate tables.
- If possible, compress larger attribute values.

- Store objects larger than 400KB in S3 and use pointers (S3 Object ID) in DynamoDB.

4.4 Integrations

- ElastiCache can be used in front of DynamoDB for performance of reads on infrequently changed data.
- Triggers integrate with AWS Lambda to respond to triggers.
- **Integration with RedShift:**
  - RedShift complements DynamoDB with advanced business intelligence.
  - When copying data from a DynamoDB table into RedShift you can perform complex data analysis queries including joins with other tables.
  - A copy operation from a DynamoDB table counts against the table's read capacity.
  - After data is copied, SQL queries do not affect the data in DynamoDB.
- **DynamoDB is integrated with Apache Hive on EMR. Hive can allow you to:**
  - Read and write data in DynamoDB tables allowing you to query DynamoDB data using a SQL-like language (HiveQL).
  - Copy data from a DynamoDB table to an S3 bucket and vice versa.
  - Copy data from a DynamoDB table into HDFS and vice versa.
  - Perform join operations on DynamoDB tables.

4.5 Scalability

- Push button scaling without downtime.
- You can scale down only 4 times per calendar day.
- AWS places some default limits on the throughput you can provision.
- These are the limits unless you request a higher amount:

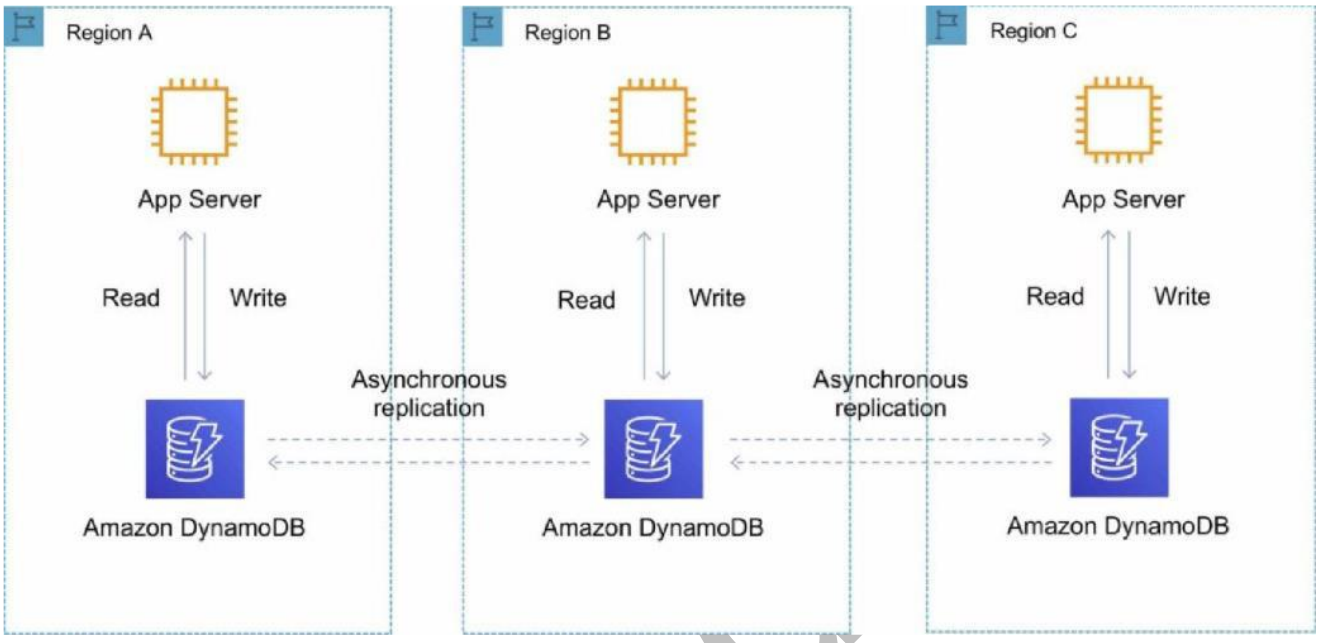
	On-Demand	Provisioned
Per table	40,000 read request units and 40,000 write request units	40,000 read capacity units and 40,000 write capacity units
Per account	Not applicable	80,000 read capacity units and 80,000 write capacity units
Minimum throughput for any table or global secondary index	Not applicable	1 read capacity unit and 1 write capacity unit

- DynamoDB can throttle requests that exceed the provisioned throughput for a table.
- DynamoDB can also throttle read requests for an Index to prevent your application from consuming too many capacity units.
- When a request is throttled it fails with an HTTP 400 code (Bad Request) and a ProvisionedThroughputExceeded exception.

4.6 Cross Region Replication With Global Tables

- Amazon DynamoDB global tables provide a fully managed solution for deploying a multi-region, multi-master database.
- When you create a global table, you specify the AWS regions where you want the table to be available.
- DynamoDB performs all of the necessary tasks to create identical tables in these regions, and propagate ongoing data changes to all of them.
- DynamoDB global tables are ideal for massively scaled applications, with globally dispersed users.
- Global tables provide automatic multi-master replication to AWS regions world-wide, so you can deliver low-latency data access to your users no matter where they are located.
- **Definitions:**
  - A global table is a collection of one or more replica tables, all owned by a single AWS account.
  - A replica table (or replica, for short) is a single DynamoDB table that functions as a part of a global table. Each replica stores the same set of data items. Any given global table can only have one replica table per region.

The following diagram depicts the Amazon DynamoDB Global Tables topology:



- You can add replica tables to the global table, so that it can be available in additional AWS regions.
- With a global table, each replica table stores the same set of data items. DynamoDB does not support partial replication of only some of the items.
- An application can read and write data to any replica table. If your application only uses eventually consistent reads, and only issues reads against one AWS region, then it will work without any modification.
- However, if your application requires strongly consistent reads, then it must perform all of its strongly consistent reads and writes in the same region. DynamoDB does not support strongly consistent reads across AWS regions.
- It is important that each replica table and secondary index in your global table has identical write capacity settings to ensure proper replication of data.

4.7 DynamoDB Auto Scaling

- DynamoDB auto scaling uses the AWS Application Auto Scaling service to dynamically adjust provisioned throughput capacity on your behalf, in response to actual traffic patterns.
- This enables a table or a global secondary index to increase its provisioned read and write capacity to handle sudden increases in traffic, without throttling.

- When the workload decreases, Application Auto Scaling decreases the throughput so that you don't pay for unused provisioned capacity.
- **How Application Auto Scaling works:**
  - You create a scaling policy for a table or a global secondary index.
  - The scaling policy specifies whether you want to scale read capacity or write capacity (or both), and the minimum and maximum provisioned capacity unit settings for the table or index.
  - The scaling policy also contains a target utilization—the percentage of consumed provisioned throughput at a point in time.
  - Uses a target tracking algorithm to adjust the provisioned throughput of the table (or index) upward or downward in response to actual workloads, so that the actual capacity utilization remains at or near your target utilization.
  - Currently, Auto Scaling does not scale down your provisioned capacity if your table's consumed capacity becomes zero.
  - If you use the AWS Management Console to create a table or a global secondary index, DynamoDB auto scaling is enabled by default.

4.8 Limits

- 256 tables per account per region.
- No limit on the size of a table.
- Read/write capacity unit limits vary per region.

4.9 Capacity Units

- One read capacity unit represents one strongly consistent read per second, or two eventually consistent reads per second for items up to 4KB.
- For items larger than 4KB, DynamoDB consumes additional read capacity units.
- One write capacity unit represents one write per second for an item up to 1KB.

4.10 Charges

- DynamoDB charges for reading, writing, and storing data in your DynamoDB tables, along with any optional features you choose to enable.
- **There are two pricing models for DynamoDB:**
  - On-demand capacity mode: DynamoDB charges you for the data reads and writes your application performs on your tables. You do not need to specify how much read and write throughput you expect your application to perform because DynamoDB instantly accommodates your workloads as they ramp up or down.

  - Provisioned capacity mode: you specify the number of reads and writes per second that you expect your application to require. You can use auto scaling to automatically adjust your table's capacity based on the specified utilization rate to ensure application performance while reducing cost.

- **Additional charges include:**
  - Data transfer out
  - Backups per GB (continuous or on-demand)
  - Global Tables
  - DynamoDB Accelerator (DAX)
  - DynamoDB Streams

4.11 High Availability Approaches For Databases

- If possible, choose DynamoDB over RDS because of inherent fault tolerance.
- If DynamoDB can't be used, choose Aurora because of redundancy and automatic recovery features.
- If Aurora can't be used, choose Multi-AZ RDS.
- Frequent RDS snapshots can protect against data corruption or failure and they won't impact performance of Multi-AZ deployment.
- Regional replication is also an option but will not be strongly consistent.
- If the database runs on EC2, you have to design the HA yourself.

4.12 Sample Questions

Q1: For which of the following are you not responsible for security?

A. DynamoDB

B. Operating system configuration

C. Server-side encryption

D. Application keys

Answer: A

Explanation: AWS manages DynamoDB as a managed service. All the other options are your responsibility as a customer of AWS.

Q2 : You are building an application that will collect information about user behavior. The application will rapidly ingest large amounts of dynamic data and requires very low latency. The database must be scalable without incurring downtime. Which database would you recommend for this scenario?

A. RDS with Microsoft SQL

B. RedShift

C. DynamoDB

D. RDS with MySQL

Answer: C

Explanation: Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. Push button scaling means that you can scale the DB at any time without incurring downtime. DynamoDB provides low read and write latency.

For more Questions Please check Certification Sample Quiz under each module

Link: https://k21academy.com/awssaquizm08

5 AMAZON ELASTICACHE

- Fully managed implementations of two popular in-memory data stores – Redis and Memcached.
- ElastiCache is a web service that makes it easy to deploy and run Memcached or Redis protocol-compliant server nodes in the cloud.
- The in-memory caching provided by ElastiCache can be used to significantly improve latency and throughput for many read-heavy application workloads or compute-intensive workloads.
- Best for scenarios where the DB load is based on Online Analytics Processing (OLAP) transactions.
- Push-button scalability for memory, writes and reads.
- In-memory key/value store – not persistent in the traditional sense.
- Billed by node size and hours of use.
- Elasticache EC2 nodes cannot be accessed from the Internet, nor can they be accessed by EC2 instances in other VPCs.
- Cached information may include the results of I/O-intensive database queries or the results of computationally-intensive calculations.
- Can be on-demand or reserved instances too (but not Spot instances).
- Elasticache can be used for storing session state.
- A node is a fixed-sized chunk of secure, network-attached RAM and is the smallest building block.
- Each node runs an instance of the Memcached or Redis protocol-compliant service and has its own DNS name and port.
- Failed nodes are automatically replaced.
- Access to Elasticache nodes is controlled by VPC security groups and subnet groups (when deployed in a VPC).
- Subnet groups are a collection of subnets designated for your Amazon ElastiCache Cluster.
- You cannot move an existing Amazon ElastiCache Cluster from outside VPC into a VPC.
- You need to configure subnet groups for Elasticache for the VPC that hosts the EC2 instances and the Elasticache cluster.
- When not using a VPC, Amazon ElastiCache allows you to control access to your clusters through Cache Security Groups (you need to link the corresponding EC2 Security Groups).

- Elasticache nodes are deployed in clusters and can span more than one subnet of the same subnet group.
- A cluster is a collection of one or more nodes using the same caching engine.
- Applications connect to Elasticache clusters using endpoints.
- An endpoint is a node or cluster's unique address.
- Maintenance windows can be defined and allow software patching to occur.
- **There are two types of ElastiCache engine:**
  - Memcached – simplest model, can run large nodes with multiple cores/threads, can be scaled in and out, can cache objects such as DBs.
  - Redis – complex model, supports encryption, master / slave replication, cross AZ (HA), automatic failover and backup/restore.

5.1 Use Cases

The following table describes a few typical use cases for ElastiCache:

Use Case	Benefit
Web session store	In cases with load-balanced web servers, store web session information in Redis so if a server is lost, the session info is not lost and another web server can pick it up
Database caching	Use Memcached in front of AWS RDS to cache popular queries to offload work from RDS and return results faster to users
Leaderboards	Use Redis to provide a live leaderboard for millions of users of your mobile app
Streaming data dashboards	Provide a landing spot for streaming sensor data on the factory floor, providing live real-time dashboard displays

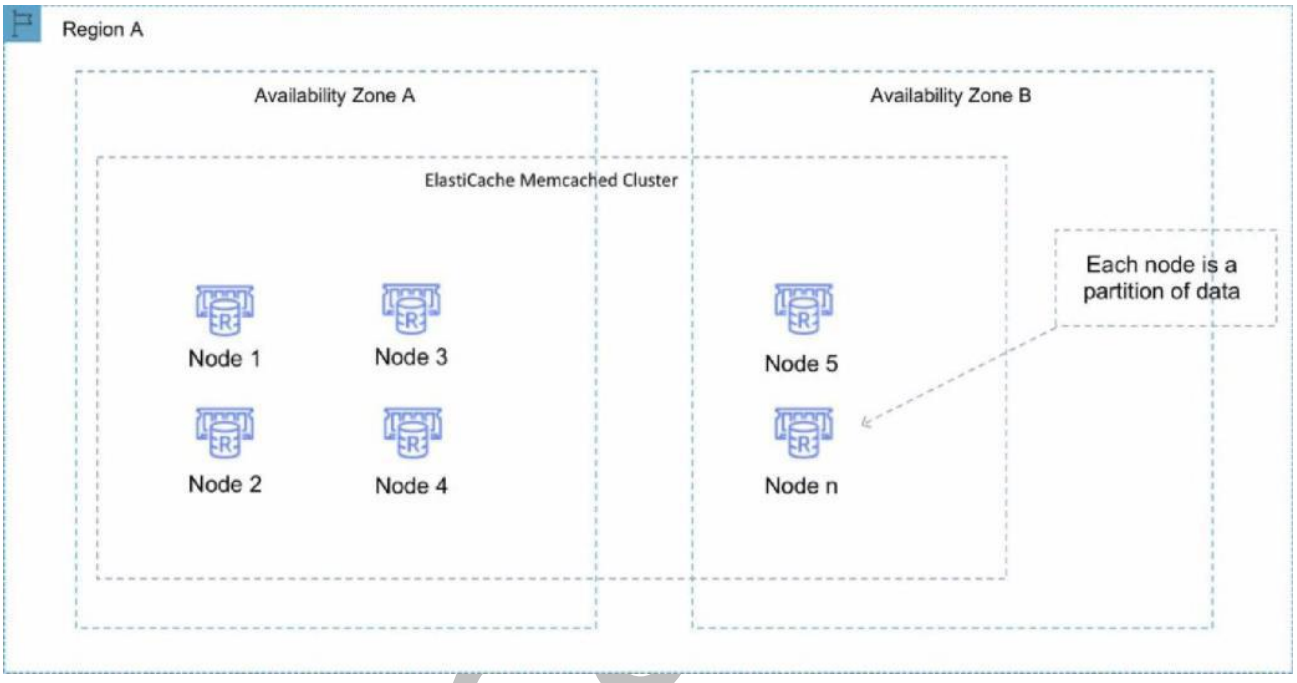
The table below describes the requirements that would determine whether to use the Memcached or Redis engine:

Memcached	Redis
Simple, no-frills	You need encryption
You need to scale-out and in as demand changes	You need HIPAA compliance
You need to run multiple CPU cores and threads	Support for clustering
You need to cache objects (e.g. database queries)	You need complex data types
	You need HA (replication
	Pub/Sub capability
	Geospacial Indexing
	Backup and restore

5.1.1 MEMCACHED

- Not persistent.
- Cannot be used as a data store.
- Supports large nodes with multiple cores or threads.
- Scales out and in, by adding and removing nodes.
- Ideal front-end for data stores (RDS, Dynamo DB etc.).
- **Use cases:**
  - Cache the contents of a DB.
  - Cache data from dynamically generated web pages.
  - Transient session data.
  - High frequency counters for admission control in high volume web apps.
- Max 100 nodes per region, 1-20 nodes per cluster (soft limits).
- Can integrate with SNS for node failure/recovery notification.
- Supports auto-discovery for nodes added/removed from the cluster.
- Scales out/in (horizontally) by adding/removing nodes.
- Scales up/down (vertically) by changing the node family/type.
- Does not support multi-AZ failover or replication.

- Does not support snapshots.
- You can place nodes in different AZs.
- With ElastiCache Memcached each node represents a partition of data and nodes in a cluster can span availability zones:

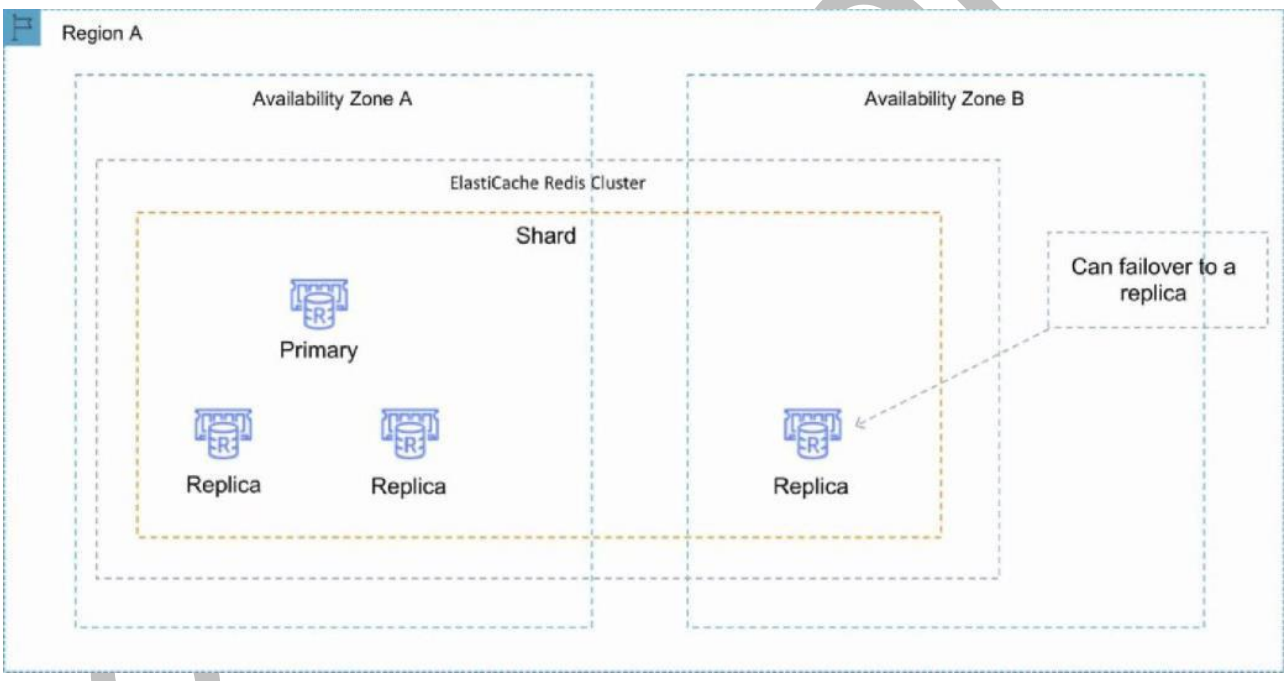


5.1.2 REDIS

- Data is persistent.
- Can be used as a datastore.
- Not multi-threaded.
- Scales by adding shards, not nodes.
- A Redis shard is a subset of the cluster's keyspace, that can include a primary node and zero or more read-replicas.
- Supports automatic and manual snapshots (S3).
- Backups include cluster data and metadata.
- You can restore your data by creating a new Redis cluster and populating it from a backup.
- Supports master/slave replication.
- During backup you cannot perform CLI or API operations on the cluster.
- Automated backups are enabled by default (automatically deleted with Redis deletion).

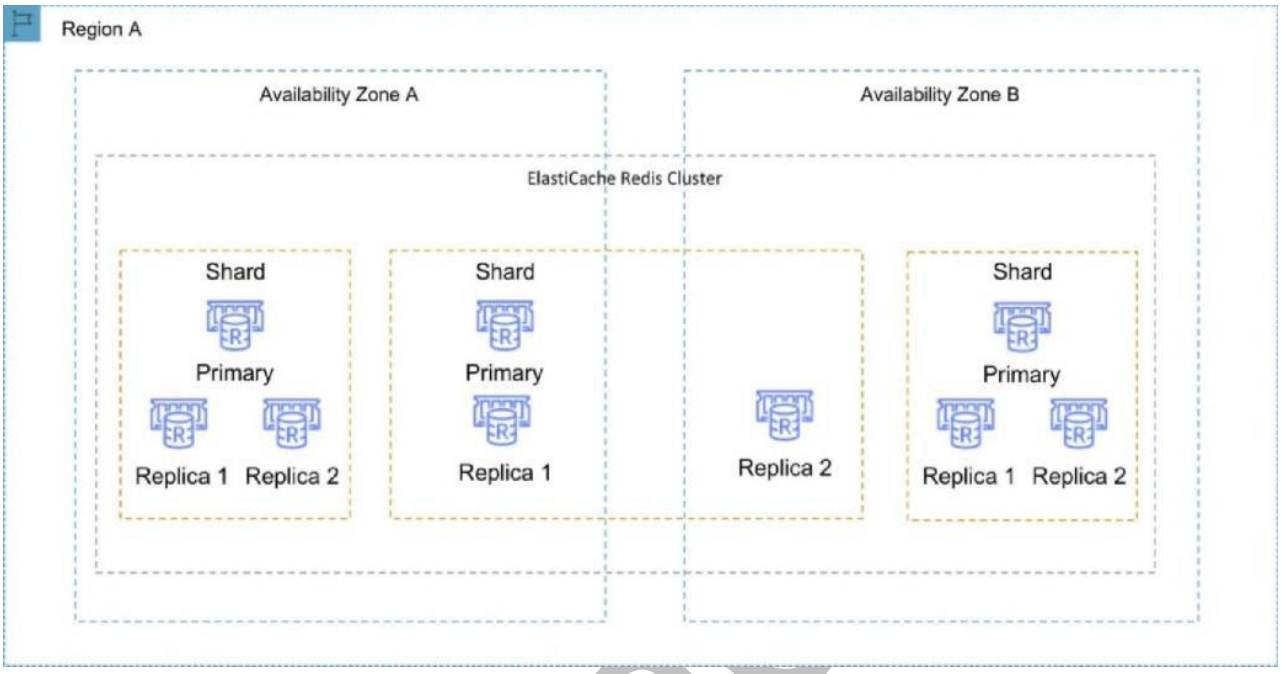
- You can only move snapshots between regions by exporting them from Elasticache before moving between regions (can then populate a new cluster with data).
- Multi-AZ is possible using read replicas in another AZ in the same region.
- **Clustering mode disabled:**
  - ➢ You can only have one shard.
  - ➢ One shard can have one read/write primary node and 0-5 read only replicas.
  - ➢ You can distribute the replicas over multiple AZs in the same region.
  - ➢ Replication from the primary node is asynchronous.

- **A Redis cluster with cluster mode disabled is represented in the diagram below:**



- **Clustering mode enabled:**
  - ➢ Can have up to 15 shards.
  - ➢ Each shard can have one primary node and 0-5 read only replicas.
  - ➢ Taking snapshots can slow down nodes, best to take from the read replicas.

- **A Redis cluster with cluster mode enabled is represented in the diagram below:**



- **Multi-AZ failover:**
  - Failures are detected by Elasticache.
  - Elasticache automatically promotes the replica that has the lowest replica lag.
  - DNS records remain the same but point to the IP of the new primary.
  - Other replicas start to sync with the new primary.
- You can have a fully automated, fault tolerant Elasticache-Redis implementation by enabling both cluster mode and multi-AZ failover.
- The following table compares the Memcached and Redis engines:

	Memcached	Redis (cluster mode disabled)	Redis (cluster mode enabled)
Engine versions	1.4.x	2.8.x and 3.2.x	3.2.x
Data types	Simple	Complex	Complex
Data partitioning	Yes	No	Yes
Cluster is modifiable	Yes	Yes	No
Online re-sharding	No	No	3.2.10
Encryption	No	3.2.6	3.2.6
HIPAA Compliance	No	3.2.6	3.2.6
Multi-threaded	Yes	No	No
Node type upgrade	No	Yes	No
Engine upgrading	Yes	Yes	No
High availability (replication)	No	Yes	Yes
Automatic failover	No	Optional	Required
Pub/Sub capabilities	No	Yes	Yes
Sorted sets	No	Yes	Yes
Backup and restore	No	Yes	Yes
Geospatial indexing	No	Yes	Yes

5.2 Charges

- Pricing is per Node-hour consumed for each Node Type.
- Partial Node-hours consumed are billed as full hours.
- There is no charge for data transfer between Amazon EC2 and Amazon Elasticache within the same Availability Zone.

5.3 High Availiability For Elasticache

Memcached:

- Because Memcached does not support replication, a node failure will result in data loss.
- Use multiple nodes in each shard to minimize data loss on node failure.
- Launch multiple nodes across available AZs to minimize data loss on AZ failure.

Redis:

- Use multiple nodes in each shard and distribute the nodes across multiple AZs.
- Enable Multi-AZ on the replication group to permit automatic failover if the primary nodes fails.
- Schedule regular backups of your Redis cluster.

5.4 Sample Questions

Q1: Your company is starting to use AWS to host new web-based applications. A new two-tier application will be deployed that provides customers with access to data records. It is important that the application is highly responsive and retrieval times are optimized. You're looking for a persistent data store that can provide the required performance. From the list below what AWS service would you recommend for this requirement?

A. RDS in a multi-AZ configuration

B. Kinesis Data Streams

C. ElastiCache with the Redis engine

D. ElastiCache with the Memcached engine

Answer: C

Explanation: ElastiCache is a web service that makes it easy to deploy and run Memcached or Redis protocol-compliant server nodes in the cloud. The in-memory caching provided by ElastiCache can be used to significantly improve latency and throughput for many read-heavy application workloads or compute-intensive workloads.

Q2 : The database layer of an on-premises web application is being migrated to AWS. The database uses a multi-threaded, in-memory caching layer to improve performance for repeated queries. Which service would be the most suitable replacement for the database cache?

A. Amazon RDS MySQL

B. Amazon ElastiCache Redis

C. Amazon DynamoDB DAX

D. Amazon ElastiCache Memcached

Answer: D

Explanation: Amazon ElastiCache with the Memcached engine is an in-memory database that can be used as a database caching layer. The memached engine supports multiple cores and threads and large nodes..

For more Questions Please check Certification Sample Quiz under each module

Link: https://k21academy.com/awssaquizm08

6 AMAZON REDSHIFT

- Amazon Redshift is a fast, fully managed data warehouse that makes it simple and cost-effective to analyze all your data using standard SQL and existing Business Intelligence (BI) tools.
- Clustered peta-byte scale data warehouse.
- RedShift is a SQL based data warehouse used for analytics applications.
- RedShift is an Online Analytics Processing (OLAP) type of DB.
- RedShift is used for running complex analytic queries against petabytes of structured data, using sophisticated query optimization, columnar storage on high-performance local disks, and massively parallel query execution.
- RedShift is ideal for processing large amounts of data for business intelligence.
- Extremely cost-effective as compared to some other on-premises data warehouse platforms.
- PostgreSQL compatible with JDBC and ODBC drivers available; compatible with most Business Intelligence tools out of the box.
- Features parallel processing and columnar data stores which are optimized for complex queries.
- Option to query directly from data files on S3 via RedShift Spectrum.
- RedShift is 10x faster than a traditional SQL DB.
- RedShift can store huge amounts of data but cannot ingest huge amounts of data in real time.
- **RedShift uses columnar data storage:**
  - Data is stored sequentially in columns instead of rows.
  - Columnar based DB is ideal for data warehousing and analytics.
  - Requires fewer I/Os which greatly enhances performance.
- **RedShift provides advanced compression:**
  - Data is stored sequentially in columns which allows for much better performance and less storage space.
  - RedShift automatically selects the compression scheme.
- RedShift provides good query performance and compression.
- RedShift provides Massively Parallel Processing (MPP) by distributing data and queries across all nodes.

- RedShift uses EC2 instances so you need to choose your instance type/size for scaling compute vertically, but you can also scale horizontally by adding more nodes to the cluster.
- You cannot have direct access to your AWS RedShift cluster nodes as a user, but you can through applications HDD and SSD storage options.
- The size of a single node is 160GB and clusters can be created up to a petabyte or more.
- Multi-node consists of:
  1. Leader node:
     - Manages client connections and receives queries.
     - Simple SQL end-point.
     - Stores metadata.
     - Optimizes query plan.
     - Coordinates query execution.
  2. Compute nodes:
     - Stores data and performs queries and computations.
     - Local columnar storage.
     - Parallel/distributed execution of all queries, loads, backups, restores, resizes.
     - Up to 128 compute nodes.
- Amazon RedShift Spectrum is a feature of Amazon Redshift that enables you to run queries against exabytes of unstructured data in Amazon S3, with no loading or ETL required.

6.1 Availability & Durability

- RedShift uses replication and continuous backups to enhance availability and improve durability and can automatically recover from component and node failures.
- Only available in one AZ but you can restore snapshots into another AZ.
- Alternatively, you can run data warehouse clusters in multiple AZ's by loading data into two Amazon Redshift data warehouse clusters in separate AZs from the same set of Amazon S3 input files.
- Redshift replicates your data within your data warehouse cluster and continuously backs up your data to Amazon S3.
- RedShift always keeps three copies of your data:
  - The original
  - A replica on compute nodes (within the cluster)

- A backup copy on S3

- **RedShift provides continuous/incremental backups:**
  - Multiple copies within a cluster.
  - Continuous and incremental backups to S3.
  - Continuous and incremental backups across regions.
  - Streaming restore.
- **RedShift provides fault tolerance for the following failures:**
  - Disk failures.
  - Nodes failures.
  - Network failures.
  - AZ/region level disasters.
- For nodes failures the data warehouse cluster will be unavailable for queries and updates until a replacement node is provisioned and added to the DB.
- **High availability for RedShift:**
  - Currently, RedShift does not support Multi-AZ deployments.
  - The best HA option is to use multi-node cluster which supports data replication and node recovery.
  - A single node RedShift cluster does not support data replication and you'll have to restore from a snapshot on S3 if a drive fails.
- RedShift can asynchronously replicate your snapshots to S3 in another region for DR.
- Single-node clusters do not support data replication (in a failure scenario you would need to restore from a snapshot).
- Scaling requires a period of unavailability of a few minutes (typically during the maintenance window).
- During scaling operations RedShift moves data in parallel from the compute nodes in your existing data warehouse cluster to the compute nodes in your new cluster.
- By default, Amazon Redshift retains backups for 1 day. You can configure this to be as long as 35 days.
- If you delete the cluster you can choose to have a final snapshot taken and retained.
- Manual backups are not automatically deleted when you delete a cluster.

6.2 Security

- You can load encrypted data from S3.
- Supports SSL Encryption in-transit between client applications and Redshift data warehouse cluster.
- VPC for network isolation.
- Encryption for data at rest (AES 256).
- Audit logging and AWS CloudTrail integration.
- RedShift takes care of key management or you can manage your own through HSM or KMS.

6.3 Charges

- Charged for compute nodes hours, 1 unit per hour (only compute node, not leader node).
- Backup storage – storage on S3.
- Data transfer – no charge for data transfer between RedShift and S3 within a region but for other scenarios you may pay charges.

6.4 Sample Questions

Q1: The database layer of an on-premises web application is being migrated to AWS. The database currently uses an in-memory cache. A Solutions Architect must deliver a solution that supports high availability and replication for the caching layer.

Which service should the Solutions Architect recommend?

A. Amazon RDS Multi-AZ

B. Amazon ElastiCache Memcached

C. Amazon ElastiCache Redis

D. Amazon DynamoDB

Answer: C

Explanation: Amazon ElastiCache Redis is an in-memory database cache and supports high availability through replicas and multi-AZ. The table below compares ElastiCache Redis with Memcached:

Q2 : A gaming company collects real-time data and stores it in an on-premises database system. The company are migrating to AWS and need better performance for the database. A solutions architect has been asked to recommend an in-memory database that supports data replication.

Which database should a solutions architect recommend?

A. Amazon ElastiCache for Memcached

B. Amazon RDS for MySQL

C. Amazon RDS for PostgreSQL

D. Amazon ElastiCache for Redis

Answer: D

Explanation: Amazon ElastiCache is an in-memory database. With ElastiCache Memcached there is no data replication or high availability.

For more Questions Please check Certification Sample Quiz under each module

Link: https://k21academy.com/awssaquizm08