**Amazon Relational Database Service (Amazon RDS)** is a managed service that you can use to launch and manage relational databases on AWS.

Amazon RDS is an Online Transaction Processing (OLTP) type of database.

The primary use case is a transactional database (rather than an analytical database).

It is best suited to structured, relational data store requirements.

It aims to be drop-in replacement for existing on-premises instances of the same databases.

Automated backups and patching are 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 exception to the above rule is Amazon RDS Custom which allows access to the underlying operating system. This is new, available for limited DB engines, and does not appear on the exam yet.

The Amazon RDS managed 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 licenses).

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-premises 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.

**Use Cases, Alternatives and 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 |

**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.

**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.

**Billing and 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 licenses 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.

**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)

**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 | 1,000-80,000 IOPS | 100 GiB-64TiB |
| SQL Server | 1,000-64,000 IOPS | 20 GiB-16TiB |
| MySQL | 1,000-80,000 IOPS | 100 GiB-64TiB |
| Oracle | 1,000-256,000 IOPS | 100 GiB-64TiB |
| PostgreSQL | 1,000-80,000 IOPS | 100 GiB-64TiB |

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.

**Multi-AZ and 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 |

**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.

The secondary DB in a multi-AZ configuration cannot be used as an independent read node (read or write).

There is no charge for data transfer between primary and secondary RDS instances.

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, MariaDB, PostgreSQL, and Oracle 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.

The process for implementing maintenance activities is as follows:

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

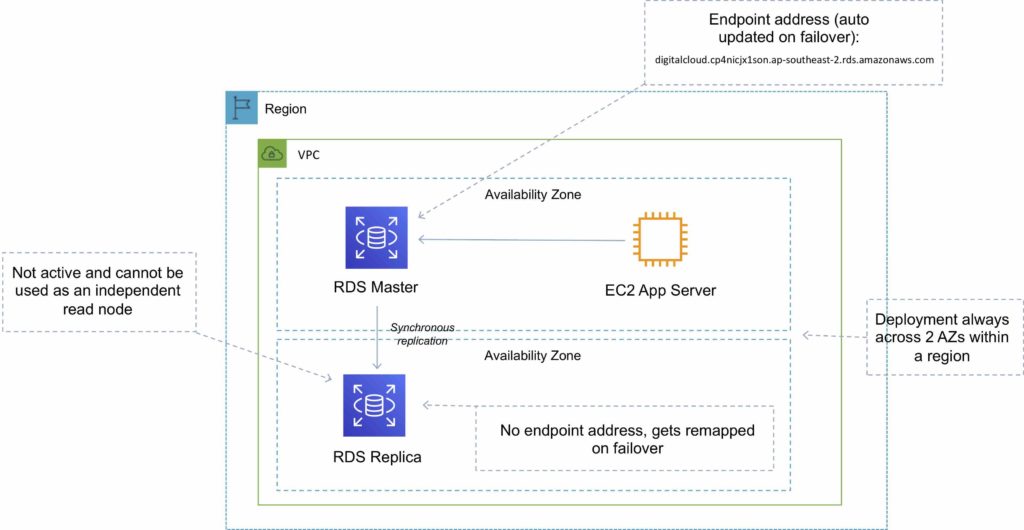
You can manually upgrade a DB instance to a supported DB engine version from the AWS Console.

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.



**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 InnoDB).

Read replicas are available for MySQL, PostgreSQL, MariaDB, Oracle, Aurora, and 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](https://aws.amazon.com/rds/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.

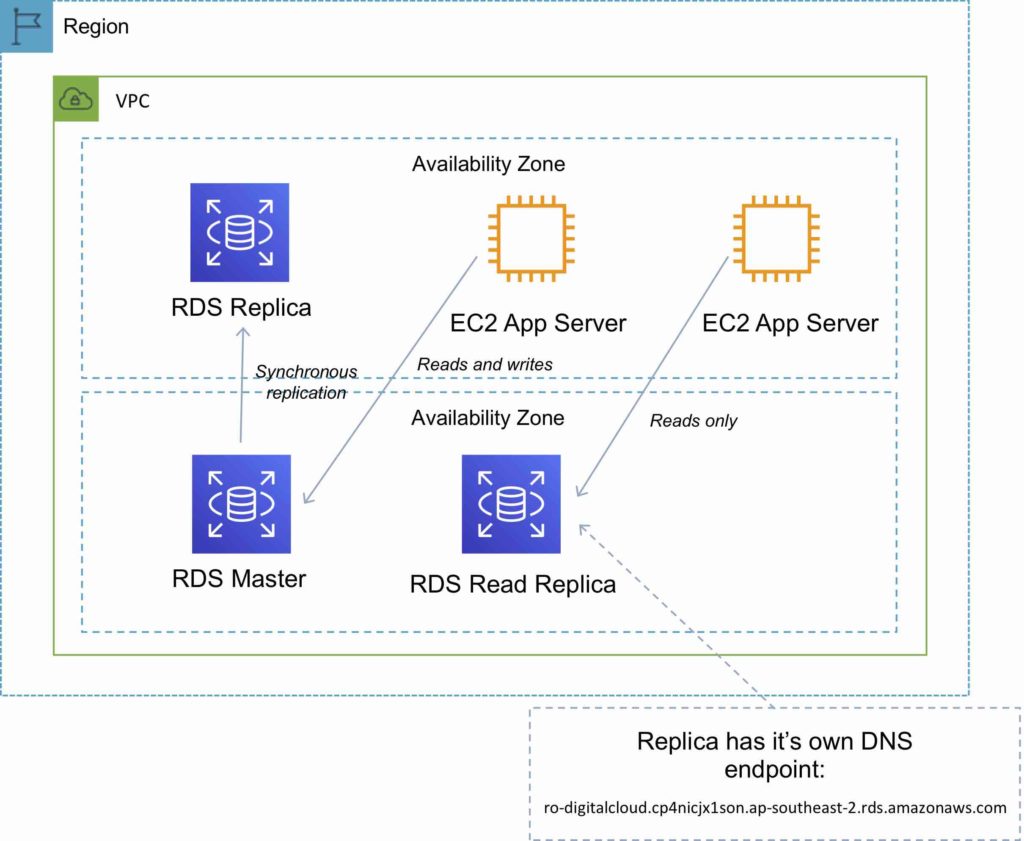
Existing read replicas continue to function as normal.

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.



**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.

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.

**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 must design the HA yourself.

**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 supports MongoDB and DynamoDB.

SCT is used for larger, more complex datasets like data warehouses.

DMS has replication functions for on-premises to AWS or to Snowball or S3.

**Monitoring, Logging and Reporting**

You can use the following automated monitoring tools to watch Amazon RDS and report when something is wrong:

* **Amazon RDS Events** – Subscribe to Amazon RDS events to be notified when changes occur with a DB instance, DB snapshot, DB parameter group, or DB security group.
* **Database log files** – View, download, or watch database log files using the Amazon RDS console or Amazon RDS API operations. You can also query some database log files that are loaded into database tables.
* **Amazon RDS Enhanced Monitoring** — Look at metrics in real time for the operating system.
* **Amazon RDS Performance Insights** — Assess the load on your database and determine when and where to act.
* **Amazon RDS Recommendations** — Look at automated recommendations for database resources, such as DB instances, read replicas, and DB parameter groups.

In addition, Amazon RDS integrates with Amazon CloudWatch, Amazon EventBridge, and AWS CloudTrail for additional monitoring capabilities:

* **Amazon CloudWatch Metrics** – Amazon RDS automatically sends metrics to CloudWatch every minute for each active database. You don’t get additional charges for Amazon RDS metrics in CloudWatch.
* **Amazon CloudWatch Alarms** – You can watch a single Amazon RDS metric over a specific time period. You can then perform one or more actions based on the value of the metric relative to a threshold that you set.
* **Amazon CloudWatch Logs** – Most DB engines enable you to monitor, store, and access your database log files in CloudWatch Logs.
* **Amazon CloudWatch Events and Amazon EventBridge** – You can automate AWS services and respond to system events such as application availability issues or resource changes. Events from AWS services are delivered to CloudWatch Events and EventBridge nearly in real time. You can write simple rules to indicate which events interest you and what automated actions to take when an event matches a rule
* **AWS CloudTrail** – You can view a record of actions taken by a user, role, or an AWS service in Amazon RDS. CloudTrail captures all API calls for Amazon RDS as events. These captures include calls from the Amazon RDS console and from code calls to the Amazon RDS API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon RDS. If you don’t configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**.

**Authorization and Access Control**

Amazon RDS supports [identity-based policies](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/security_iam_service-with-iam.html#security_iam_service-with-iam-resource-based-policies).

RDS does not support resource-based policies.

The following AWS managed policies, which you can attach to users in your account, are specific to Amazon RDS:

* **AmazonRDSReadOnlyAccess** – Grants read-only access to all Amazon RDS resources for the AWS account specified.
* **AmazonRDSFullAccess** – Grants full access to all Amazon RDS resources for the AWS account specified.

You can authenticate to your DB instance using AWS Identity and Access Management (IAM) database authentication. IAM database authentication works with MySQL and PostgreSQL. With this authentication method, you don’t need to use a password when you connect to a DB instance. Instead, you use an authentication token.

IAM database authentication provides the following benefits:

* Network traffic to and from the database is encrypted using Secure Sockets Layer (SSL).
* You can use IAM to centrally manage access to your database resources, instead of managing access individually on each DB instance.
* For applications running on Amazon EC2, you can use profile credentials specific to your EC2 instance to access your database instead of a password, for greater security.

<https://digitalcloud.training/aws-secrets-manager/>

<https://digitalcloud.training/aws-resource-access-manager/>

<https://digitalcloud.training/amazon-s3-and-glacier/>