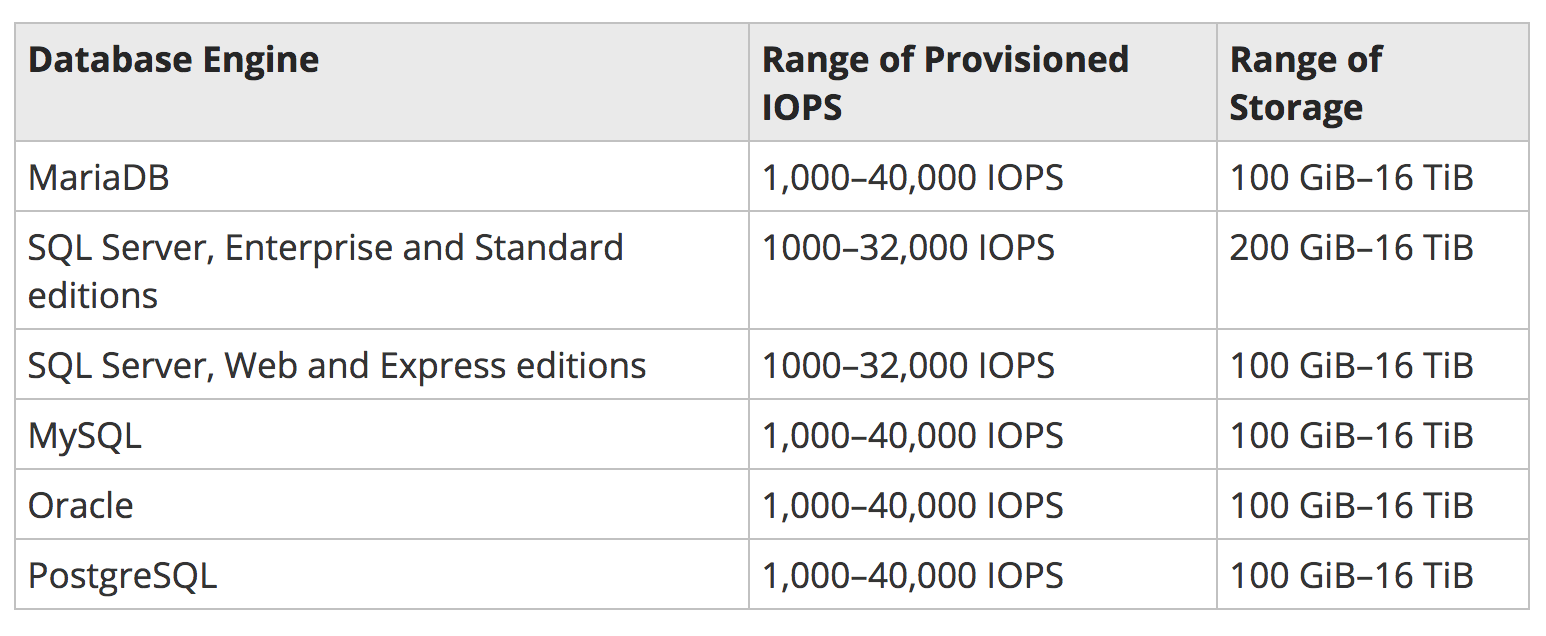
[Amazon Aurora](https://aws.amazon.com/rds/aurora/) is a MySQL and PostgreSQL-compatible [relational database](https://aws.amazon.com/relational-database/) engine that combines the speed and availability of high-end commercial databases with the simplicity and cost-effectiveness of open source databases. Amazon Aurora provides up to five times better performance than MySQL and up to three times better performance than PostgreSQL with the security, availability, and reliability of a commercial database at one-tenth the cost.

[AWS Database Migration Services (DMS)](https://aws.amazon.com/dms/) helps you migrate databases to AWS easily and securely. 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. The service supports homogenous migrations such as Oracle to RDS for Oracle, as well as heterogeneous migrations between different database platforms, such as Oracle to Amazon Aurora or Microsoft SQL Server to RDS for MySQL.

For production application that requires fast and consistent I/O performance, we recommend Provisioned IOPS (input/output operations per second) storage. Provisioned IOPS storage is a storage type that delivers predictable performance, and consistently low latency. Provisioned IOPS storage is optimized for online transaction processing (OLTP) workloads that have consistent performance requirements. Provisioned IOPS helps performance tuning of these workloads.

For production OLTP use cases, we recommend that you use Multi-AZ deployments for enhanced fault tolerance with Provisioned IOPS storage for fast and predictable performance.

You can also use Provisioned IOPS SSD storage with Read Replicas for MySQL, MariaDB or PostgreSQL. The type of storage for a Read Replica is independent of that on the master DB instance.



Amazon Elastic Block Store (Amazon EBS) provides persistent block storage volumes for use with [Amazon EC2](https://aws.amazon.com/ec2-sla/) instances in the AWS Cloud. Each Amazon EBS volume is automatically replicated within its Availability Zone to protect you from component failure, offering high availability and durability. Amazon EBS volumes offer the consistent and low-latency performance needed to run your workloads. With Amazon EBS, you can scale your usage up or down within minutes – all while paying a low price for only what you provision.

Amazon EBS is designed for application workloads that benefit from fine tuning for performance, cost and capacity. Typical use cases include Big Data analytics engines (like the Hadoop/HDFS ecosystem and [Amazon EMR](https://aws.amazon.com/emr/) clusters), relational and NoSQL databases (like Microsoft SQL Server and MySQL or Cassandra and MongoDB), stream and log processing applications (like Kafka and Splunk), and data warehousing applications (like Vertica and Teradata).

Amazon EBS allows you to create storage volumes and attach them to Amazon EC2 instances. Once attached, you can create a file system on top of these volumes, run a database, or use them in any other way you would use block storage. Amazon EBS volumes are placed in a specific Availability Zone where they are automatically replicated to protect you from the failure of a single component. All EBS volume types offer durable snapshot capabilities and are designed for 99.999% availability.

Amazon EBS provides a range of options that allow you to optimize storage performance and cost for your workload. These options are divided into two major categories: SSD-backed storage for transactional workloads, such as databases and boot volumes (performance depends primarily on IOPS), and HDD-backed storage for throughput intensive workloads, such as MapReduce and log processing (performance depends primarily on MB/s).

SSD-backed volumes include the highest performance Provisioned IOPS SSD (io1) for latency-sensitive transactional workloads and General Purpose SSD (gp2) that balance price and performance for a wide variety of transactional data. HDD-backed volumes include Throughput Optimized HDD (st1) for frequently accessed, throughput intensive workloads and the lowest cost Cold HDD (sc1) for less frequently accessed data.

Elastic Volumes is a feature of Amazon EBS that allows you to dynamically increase capacity, tune performance, and change the type of live volumes with no downtime or performance impact. This allows you to easily right-size your deployment and adapt to performance changes.

I can "force" a failover for any RDS instance that has Multi-AZ configured

**General Purpose SSD** – General Purpose SSD , also called gp2, volumes offer cost-effective storage that is ideal for a broad range of workloads. These volumes deliver single-digit millisecond latencies and the ability to burst to 3,000 IOPS for extended periods of time.

**Provisioned IOPS** – Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads, that require low I/O latency and consistent I/O throughput.

**Magnetic** – Amazon RDS also supports magnetic storage for backward compatibility. We recommend that you use General Purpose SSD or Provisioned IOPS for any new storage needs.

**DB instance class**

To get the most performance out of your Amazon RDS database instance, choose a current generation instance type with enough bandwidth to support your storage type. For example, you can choose EBS-optimized instances and instances with 10-gigabit network connectivity.

Each *AWS Region* is a separate geographic area. Each AWS Region has multiple, isolated locations known as *Availability Zones*. Amazon RDS provides you the ability to place resources, such as instances, and data in multiple locations. Resources aren't replicated across AWS Regions unless you do so specifically.

The default AWS Region can be changed in the console, by setting the EC2\_REGION environment variable

In a Multi-AZ deployment, Amazon RDS automatically provisions and maintains a synchronous standby replica in a different Availability Zone. The primary DB instance is synchronously replicated across Availability Zones to a standby replica to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.

Amazon RDS sets up a secure communications channel between the source DB instance and a Read Replica if that Read Replica is in a different AWS Region from the DB instance.

* Scaling beyond the compute or I/O capacity of a single DB instance for read-heavy database workloads. You can direct this excess read traffic to one or more Read Replicas.
* Serving read traffic while the source DB instance is unavailable. If your source DB instance can't take I/O requests (for example, due to I/O suspension for backups or scheduled maintenance), you can direct read traffic to your Read Replicas. For this use case, keep in mind that the data on the Read Replica might be "stale" because the source DB instance is unavailable.
* Business reporting or data warehousing scenarios where you might want business reporting queries to run against a Read Replica, rather than your primary, production DB instance.
* Implementing disaster recovery. You can use promote a Read Replica to a standalone instances as a disaster recovery solution if the source DB instance fails.

Amazon RDS doesn't support circular replication. You can't configure a DB instance to serve as a replication source for an existing DB instance; you can only create a new Read Replica from an existing DB instance.

You can promote a MySQL, MariaDB, or PostgreSQL Read Replica into a standalone DB instance. When you promote a Read Replica, the DB instance is rebooted before it becomes available.

You can now create up to five in-region and cross-region replicas per source with a single API call or a couple of clicks in the AWS Management Console.

Your Amazon RDS backup storage for each region is composed of the automated backups and manual DB snapshots for that region. Your backup storage is equivalent to the sum of the database storage for all instances in that region.

All automated backups are deleted when you delete a DB instance. After you delete a DB instance, the automated backups can't be recovered. If you choose to have Amazon RDS create a final DB snapshot before it deletes your DB instance, you can use that to recover your DB instance.

Manual snapshots are not deleted.

If you don't specify a preferred backup window when you create the DB instance, Amazon RDS assigns a default 30-minute backup window which is selected at random from an 8-hour block of time per region.

You can set the backup retention period to between 1 and 35 days.

Amazon RDS reserved instances give you the option to reserve a DB instance for a one or three year term and in turn receive a significant discount compared to the on-demand instance pricing for the DB instance. There are three RI payment options  -- No Upfront, Partial Upfront, All Upfront

Functionally, reserved instances and on-demand DB instances are exactly the same. The only difference is how your DB instance(s) are billed: With Reserved Instances, you purchase a one or three year reservation and in return receive a lower effective hourly usage rate (compared with on-demand DB instances) for the duration of the term. Unless you purchase reserved instances in a Region, all DB instances will be billed at on-demand hourly rates.

* Run your DB instance in an Amazon Virtual Private Cloud (VPC) for the greatest possible network access control. For more information about creating a DB instance in a VPC, see [Using Amazon RDS with Amazon Virtual Private Cloud (VPC)](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_VPC.html).
* Use AWS Identity and Access Management (IAM) policies to assign permissions that determine who is allowed to manage RDS resources. For example, you can use IAM to determine who is allowed to create, describe, modify, and delete DB instances, tag resources, or modify DB security groups. For information on setting up an IAM user, see [Create an IAM User](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_SettingUp.html#CHAP_SettingUp.IAM)
* Use security groups to control what IP addresses or Amazon EC2 instances can connect to your databases on a DB instance. When you first create a DB instance, its firewall prevents any database access except through rules specified by an associated security group.
* Use Secure Socket Layer (SSL) connections with DB instances running the MySQL, Amazon Aurora, MariaDB, PostgreSQL, Oracle, or Microsoft SQL Server database engines. For more information on using SSL with a DB instance, see [Using SSL to Encrypt a Connection to a DB Instance](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.SSL.html).
* Use RDS encryption to secure your RDS instances and snapshots at rest. RDS encryption uses the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your RDS instance. For more information, see [Encrypting Amazon RDS Resources](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.Encryption.html).
* Use network encryption and transparent data encryption with Oracle DB instances; for more information, see [Oracle Native Network Encryption](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.Oracle.Options.NetworkEncryption.html) and [Oracle Transparent Data Encryption](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.Oracle.Options.AdvSecurity.html)
* Use the security features of your DB engine to control who can log in to the databases on a DB instance, just as you do if the database was on your local network.

Amazon DynamoDB provides on-demand backup capability. It allows you to create full backups of your tables for long-term retention and archival for regulatory compliance needs.

Point-in-time recovery helps protect your Amazon DynamoDB tables from accidental write or delete operations. With point in time recovery, you don't have to worry about creating, maintaining, or scheduling on-demand backups. For example, suppose that a test script writes accidentally to a production DynamoDB table. With point-in-time recovery, you can restore that table to any point in time during the last 35 days. DynamoDB maintains incremental backups of your table.