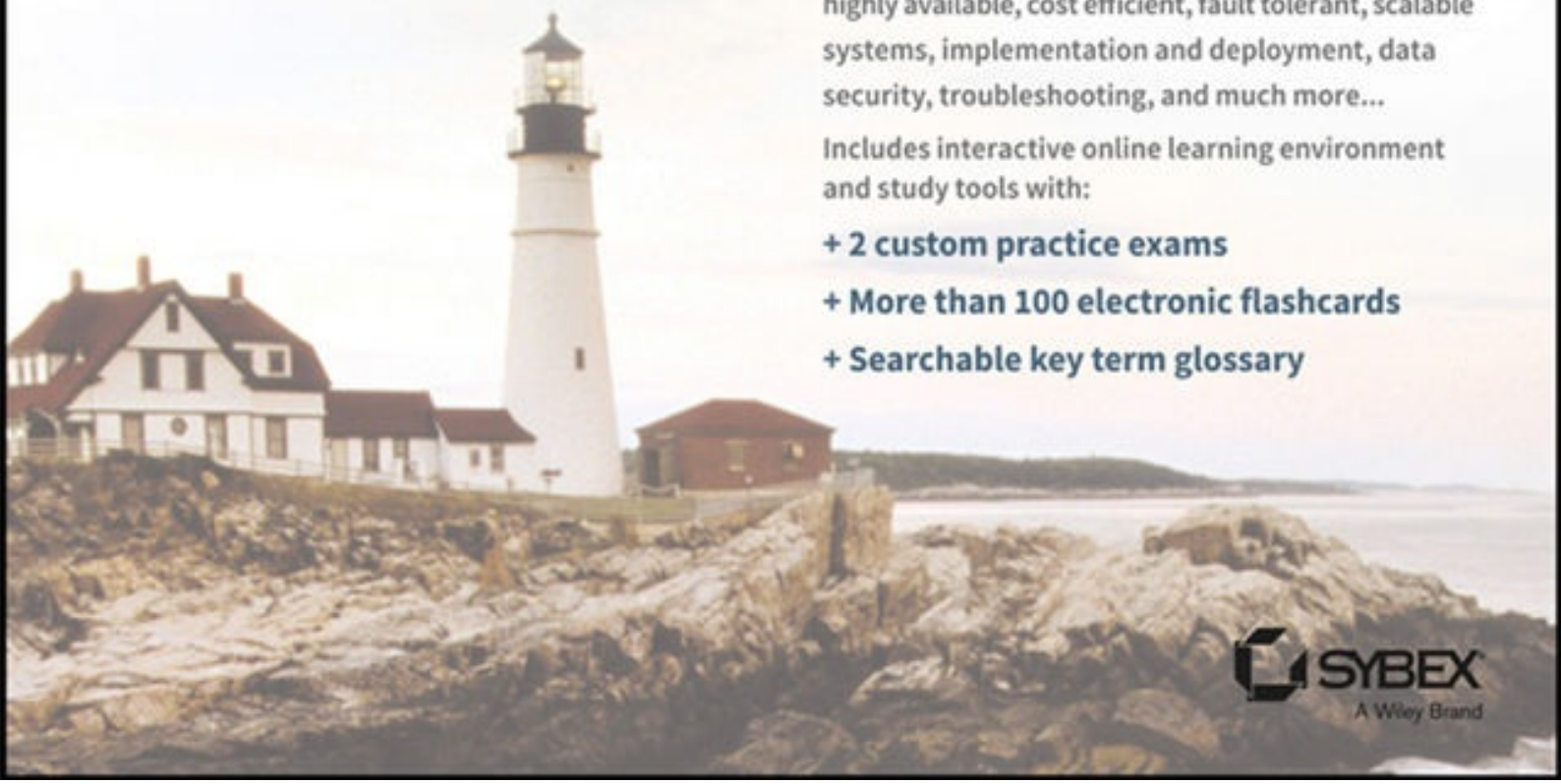
amazon web services

Joe Baron, Hisham Baz, Tim Bixler, Biff Gaut,
Kevin E. Kelly, Sean Senior, John Stamper

AWS Certified Solutions Architect

OFFICIAL STUDY GUIDE

ASSOCIATE EXAM



Covers exam objectives, including designing highly available, cost efficient, fault tolerant, scalable systems, implementation and deployment, data security, troubleshooting, and much more...

Includes interactive online learning environment and study tools with:

+ 2 custom practice exams
+ More than 100 electronic flashcards
+ Searchable key term glossary

SYBEX
A Wiley Brand

availability and durability. By delivering consistent and low-latency performance, Amazon EBS provides the disk storage needed to run a wide variety of workloads.

AWS Storage Gateway

AWS Storage Gateway is a service connecting an on-premises software appliance with cloud-based storage to provide seamless and secure integration between an organization's on-premises IT environment and the AWS storage infrastructure. The service supports industry-standard storage protocols that work with existing applications. It provides low-latency performance by maintaining a cache of frequently accessed data on-premises while securely storing all of your data encrypted in Amazon S3 or Amazon Glacier.

Amazon CloudFront

Amazon CloudFront is a content delivery web service. It integrates with other AWS Cloud services to give developers and businesses an easy way to distribute content to users across the world with low latency, high data transfer speeds, and no minimum usage commitments. Amazon CloudFront can be used to deliver your entire website, including dynamic, static, streaming, and interactive content, using a global network of edge locations. Requests for content are automatically routed to the nearest edge location, so content is delivered with the best possible performance to end users around the globe.

Database Services

AWS provides fully managed relational and NoSQL database services, and in-memory caching as a service and a petabyte-scale data warehouse solution. This section provides an overview of the products that the database services comprise.

Amazon Relational Database Service (Amazon RDS)

Amazon Relational Database Service (Amazon RDS) provides a fully managed relational database with support for many popular open source and commercial database engines. It's a cost-efficient service that allows organizations to launch secure, highly available, fault-tolerant, production-ready databases in minutes. Because Amazon RDS manages time-consuming administration tasks, including backups, software patching, monitoring, scaling, and replication, organizational resources can focus on revenue-generating applications and business instead of mundane operational tasks.

Amazon DynamoDB

Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. It is a fully managed database and supports both document and key/value data models. Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad-tech, Internet of Things, and many other applications.

Amazon Redshift

Amazon Redshift is a fast, fully managed, petabyte-scale data warehouse service that makes it simple and cost effective to analyze structured data. Amazon Redshift provides a standard SQL interface that lets organizations use existing business intelligence tools. By leveraging

Chapter 7
Databases and AWS

THE AWS CERTIFIED SOLUTIONS ARCHITECT ASSOCIATE EXAM OBJECTIVES COVERED IN THIS CHAPTER MAY INCLUDE, BUT ARE NOT LIMITED TO, THE FOLLOWING:

Domain 1.0: Designing highly available, cost-efficient, fault-tolerant, and scalable systems

✓ **1.1 Identify and recognize cloud architecture considerations, such as fundamental components and effective designs.**

Content may include the following:

- Planning and design
- Architectural trade-off decisions (Amazon Relational Database Service [Amazon RDS] vs. installing on Amazon Elastic Compute Cloud [Amazon EC2])
- Best practices for AWS architecture
- Recovery Time Objective (RTO) and Recovery Point Objective (RPO) Disaster Recovery (DR) design
- Elasticity and scalability

Domain 3.0: Data Security

✓ **3.1 Recognize and implement secure practices for optimum cloud deployment and maintenance.**

Content may include the following:

- AWS administration and security services
- Design patterns

✓ **3.2 Recognize critical disaster recovery techniques and their implementation.**

This chapter will cover essential database concepts and introduce three of Amazon's managed database services: Amazon Relational Database Service (Amazon RDS), Amazon DynamoDB, and Amazon Redshift. These managed services simplify the setup and operation of relational databases, NoSQL databases, and data warehouses.

This chapter focuses on key topics you need to understand for the exam, including:

- The differences among a relational database, a NoSQL database, and a data warehouse
- The benefits and tradeoffs between running a database on Amazon EC2 or on Amazon RDS
- How to deploy database engines into the cloud
- How to back up and recover your database and meet your Recovery Point Objective (RPO) and Recovery Time Objective (RTO) requirements
- How to build highly available database architectures
- How to scale your database compute and storage vertically
- How to select the right type of storage volume
- How to use read replicas to scale horizontally
- How to design and scale an Amazon DynamoDB table
- How to read and write from an Amazon DynamoDB table
- How to use secondary indexes to speed queries
- How to design an Amazon Redshift table
- How to load and query an Amazon Redshift data warehouse
- How to secure your databases, tables, and clusters

Database Primer

Almost every application relies on a database to store important data and records for its users. A database engine allows your application to access, manage, and search large volumes of data records. In a well-architected application, the database will need to meet the performance demands, the availability needs, and the recoverability characteristics of the system.

Database systems and engines can be grouped into two broad categories: Relational Database Management Systems (RDBMS) and NoSQL (or non-relational) databases. It is not uncommon to build an application using a combination of RDBMS and NoSQL databases. A strong understanding of essential database concepts, Amazon RDS, and Amazon DynamoDB are required to pass this exam.

Relational Databases

The most common type of database in use today is the *relational database*. The relational database has roots going back to the 1970s when Edgar F. Codd, working for IBM, developed the concepts of the relational model. Today, relational databases power all types of applications from social media apps, e-commerce websites, and blogs to complex enterprise applications. Commonly used relational database software packages include MySQL, PostgreSQL, Microsoft SQL Server, and Oracle.

Relational databases provide a common interface that lets users read and write from the database using commands or queries written using *Structured Query Language (SQL)*. A relational database consists of one or more tables, and a table consists of columns and rows similar to a spreadsheet. A database column contains a specific attribute of the record, such as a person's name, address, and telephone number. Each attribute is assigned a data type such as text, number, or date, and the database engine will reject invalid inputs.

A database row comprises an individual record, such as the details about a student who attends a school. Consider the example in Table 7.1.

TABLE 7.1 Students Table

StudentID	FirstName	LastName	Gender	Age
1001	Joe	Dusty	M	29
1002	Andrea	Romanov	F	20
1003	Ben	Johnson	M	30
1004	Beth	Roberts	F	30

This is an example of a basic table that would sit in a relational database. There are five fields with different data types:

`StudentID` = Number or integer

`FirstName` = String

`LastName` = String

`Gender` = String (Character Length = 1)

`Age` = Integer

This sample table has four records, with each record representing an individual student. Each student has a `StudentID` field, which is usually a unique number per student. A unique number that identifies each student can be called a *primary key*.

One record in a table can relate to a record in another table by referencing the primary key of a record. This pointer or reference is called a foreign key. For example, the Grades table that records scores for each student would have its own primary key and an additional column known as a foreign key that refers to the primary key of the student record. By referencing the primary keys of other tables, relational databases minimize duplication of data in associated tables. With relational databases, it is important to note that the structure of the table (such as the number of columns and data type of each column) must be defined prior to data being added to the table.

A relational database can be categorized as either an *Online Transaction Processing (OLTP)* or *Online Analytical Processing (OLAP)* database system, depending on how the tables are organized and how the application uses the relational database. OLTP refers to transaction-oriented applications that are frequently writing and changing data (for example, data entry and e-commerce). OLAP is typically the domain of data warehouses and refers to reporting or analyzing large data sets. Large applications often have a mix of both OLTP and OLAP databases.

Amazon Relational Database Service (Amazon RDS) significantly simplifies the setup and maintenance of OLTP and OLAP databases. Amazon RDS provides support for six popular relational database engines: MySQL, Oracle, PostgreSQL, Microsoft SQL Server, MariaDB, and Amazon Aurora. You can also choose to run nearly any database engine using Windows or Linux Amazon Elastic Compute Cloud (Amazon EC2) instances and manage the installation and administration yourself.

Data Warehouses

A *data warehouse* is a central repository for data that can come from one or more sources. This data repository is often a specialized type of relational database that can be used for reporting and analysis via OLAP. Organizations typically use data warehouses to compile reports and search the database using highly complex queries.

Data warehouses are also typically updated on a batch schedule multiple times per day or per hour, compared to an OLTP relational database that can be updated thousands of times per second. Many organizations split their relational databases into two different databases: one database as their main production database for OLTP transactions, and the other database as their data warehouse for OLAP. OLTP transactions occur frequently and are relatively simple. OLAP transactions occur much less frequently but are much more complex.

Amazon RDS is often used for OLTP workloads, but it can also be used for OLAP. *Amazon Redshift* is a high-performance data warehouse designed specifically for OLAP use cases. It is also common to combine Amazon RDS with Amazon Redshift in the same application and periodically extract recent transactions and load them into a reporting database.

NoSQL Databases

NoSQL databases have gained significant popularity in recent years because they are often simpler to use, more flexible, and can achieve performance levels that are difficult or impossible with traditional relational databases. Traditional relational databases are difficult to scale beyond a single server without significant engineering and cost, but a NoSQL architecture allows for horizontal scalability on commodity hardware.

NoSQL databases are non-relational and do not have the same table and column semantics of a relational database. NoSQL databases are instead often key/value stores or document stores with flexible schemas that can evolve over time or vary. Contrast that to a relational database, which requires a very rigid schema.

Many of the concepts of NoSQL architectures trace their foundational concepts back to whitepapers published in 2006 and 2007 that described distributed systems like Dynamo at Amazon. Today, many application teams use Hbase, MongoDB, Cassandra, CouchDB, Riak, and *Amazon DynamoDB* to store large volumes of data with high transaction rates. Many of these database engines support clustering and scale horizontally across many machines for performance and fault tolerance. A common use case for NoSQL is managing user session state, user profiles, shopping cart data, or time-series data.

You can run any type of NoSQL database on AWS using Amazon EC2, or you can choose a managed service like Amazon DynamoDB to deal with the heavy lifting involved with building a distributed cluster spanning multiple data centers.

Amazon Relational Database Service (Amazon RDS)

Amazon RDS is a service that simplifies the setup, operations, and scaling of a relational database on AWS. With Amazon RDS, you can spend more time focusing on the application and the schema and let Amazon RDS offload common tasks like backups, patching, scaling, and replication.

Amazon RDS helps you to streamline the installation of the database software and also the provisioning of infrastructure capacity. Within a few minutes, Amazon RDS can launch one of many popular database engines that is ready to start taking SQL transactions. After the initial launch, Amazon RDS simplifies ongoing maintenance by automating common administrative tasks on a recurring basis.

With Amazon RDS, you can accelerate your development timelines and establish a consistent operating model for managing relational databases. For example, Amazon RDS makes it easy to replicate your data to increase availability, improve durability, or scale up or beyond a single database instance for read-heavy database workloads.

Amazon RDS exposes a database endpoint to which client software can connect and execute SQL. Amazon RDS does not provide shell access to Database (DB) Instances, and it restricts access to certain system procedures and tables that require advanced privileges. With Amazon RDS, you can typically use the same tools to query, analyze, modify, and administer the database. For example, current Extract, Transform, Load (ETL) tools and reporting tools can connect to Amazon RDS databases in the same way with the same drivers, and often all it takes to reconfigure is changing the hostname in the connection string.

Database (DB) Instances

The Amazon RDS service itself provides an Application Programming Interface (API) that lets you create and manage one or more *DB Instances*. A DB Instance is an isolated database environment deployed in your private network segments in the cloud. Each DB Instance runs and manages a popular commercial or open source database engine on your behalf. Amazon RDS currently supports the following database engines: MySQL, PostgreSQL, MariaDB, Oracle, SQL Server, and Amazon Aurora.

You can launch a new DB Instance by calling the `CreateDBInstance` API or by using the AWS Management Console. Existing DB Instances can be changed or resized using the `ModifyDBInstance` API. A DB Instance can contain multiple different databases, all of which you create and manage within the DB Instance itself by executing SQL commands with the Amazon RDS endpoint. The different databases can be created, accessed, and managed using the same SQL client tools and applications that you use today.

The compute and memory resources of a DB Instance are determined by its DB Instance class. You can select the DB Instance class that best meets your needs for compute and memory. The range of DB Instance classes extends from a db.t2.micro with 1 virtual CPU (vCPU) and 1 GB of memory, up to a db.r3.8xlarge with 32 vCPUs and 244 GB of memory. As your needs change over time, you can change the instance class and the balance of compute of memory, and Amazon RDS will migrate your data to a larger or smaller instance class. Independent from the DB Instance class that you select, you can also control the size and

performance characteristics of the storage used.

Amazon RDS supports a large variety of engines, versions, and feature combinations. Check the Amazon RDS documentation to determine support for specific features. Many features and common configuration settings are exposed and managed using *DB parameter groups* and *DB option groups*. A DB parameter group acts as a container for engine configuration values that can be applied to one or more DB Instances. You may change the DB parameter group for an existing instance, but a reboot is required. A DB option group acts as a container for engine features, which is empty by default. In order to enable specific features of a DB engine (for example, Oracle Statspack, Microsoft SQL Server Mirroring), you create a new DB option group and configure the settings accordingly.


Existing databases can be migrated to Amazon RDS using native tools and techniques that vary depending on the engine. For example with MySQL, you can export a backup using mysqldump and import the file into Amazon RDS MySQL. You can also use the AWS Database Migration Service, which gives you a graphical interface that simplifies the migration of both schema and data between databases. AWS Database Migration Service also helps convert databases from one database engine to another.

Operational Benefits

Amazon RDS increases the operational reliability of your databases by applying a very consistent deployment and operational model. This level of consistency is achieved in part by limiting the types of changes that can be made to the underlying infrastructure and through the extensive use of automation. For example with Amazon RDS, you cannot use Secure Shell (SSH) to log in to the host instance and install a custom piece of software. You can, however, connect using SQL administrator tools or use DB option groups and DB parameter groups to change the behavior or feature configuration for a DB Instance. If you want full control of the Operating System (OS) or require elevated permissions to run, then consider installing your database on Amazon EC2 instead of Amazon RDS.

Amazon RDS is designed to simplify the common tasks required to operate a relational database in a reliable manner. It's useful to compare the responsibilities of an administrator when operating a relational database in your data center, on Amazon EC2, or with Amazon RDS (see Table 7.2).

TABLE 7.2 Comparison of Operational Responsibilities

Responsibility	Database On-Premise	Database on Amazon EC2	Database on Amazon RDS
App Optimization	You	You	You
Scaling	You	You	AWS
High Availability	You	You	AWS
Backups	You	You	AWS
DB Engine Patches	You	You	AWS
Software Installation	You	You	AWS
OS Patches	You	You	AWS
OS Installation	You	AWS	AWS
Server Maintenance	You	AWS	AWS
Rack and Stack	You	AWS	AWS
Power and Cooling	You	AWS	AWS

Database Engines

Amazon RDS supports six database engines: MySQL, PostgreSQL, MariaDB, Oracle, SQL Server, and Amazon Aurora. Features and capabilities vary slightly depending on the engine that you select.

MySQL

MySQL is one of the most popular open source databases in the world, and it is used to power a wide range of applications, from small personal blogs to some of the largest websites in the world. As of the time of this writing, Amazon RDS for MySQL currently supports MySQL 5.7, 5.6, 5.5, and 5.1. The engine is running the open source Community Edition with InnoDB as the default and recommended database storage engine. Amazon RDS MySQL allows you to connect using standard MySQL tools such as MySQL Workbench or SQL Workbench/J. Amazon RDS MySQL supports *Multi-AZ* deployments for high availability and *read replicas* for horizontal scaling.

PostgreSQL

PostgreSQL is a widely used open source database engine with a very rich set of features and advanced functionality. Amazon RDS supports DB Instances running several versions of PostgreSQL. As of the time of this writing, Amazon RDS supports multiple releases of PostgreSQL, including 9.5.x, 9.4.x, and 9.3.x. Amazon RDS PostgreSQL can be managed using standard tools like pgAdmin and supports standard JDBC/ODBC drivers. Amazon RDS PostgreSQL also supports Multi-AZ deployment for high availability and read replicas for

horizontal scaling.

MariaDB

Amazon RDS recently added support for DB Instances running MariaDB. MariaDB is a popular open source database engine built by the creators of MySQL and enhanced with enterprise tools and functionality. MariaDB adds features that enhance the performance, availability, and scalability of MySQL. As of the time of this writing, AWS supports MariaDB version 10.0.17. Amazon RDS fully supports the XtraDB storage engine for MariaDB DB Instances and, like Amazon RDS MySQL and PostgreSQL, has support for Multi-AZ deployment and read replicas.

Oracle

Oracle is one of the most popular relational databases used in the enterprise and is fully supported by Amazon RDS. As of the time of this writing, Amazon RDS supports DB Instances running several editions of Oracle 11g and Oracle 12c. Amazon RDS supports access to schemas on a DB Instance using any standard SQL client application, such as Oracle SQL Plus.

Amazon RDS Oracle supports three different editions of the popular database engine: Standard Edition One, Standard Edition, and Enterprise Edition. Table 7.3 outlines some of the major differences between editions:

TABLE 7.3 Amazon RDS Oracle Editions Compared

Edition	Performance	Multi-AZ	Encryption
Standard One	++++	Yes	KMS
Standard	++++++++	Yes	KMS
Enterprise	++++++++	Yes	KMS and TDE

Microsoft SQL Server

Microsoft SQL Server is another very popular relational database used in the enterprise. Amazon RDS allows Database Administrators (DBAs) to connect to their SQL Server DB Instance in the cloud using native tools like SQL Server Management Studio. As of the time of this writing, Amazon RDS provides support for several versions of Microsoft SQL Server, including SQL Server 2008 R2, SQL Server 2012, and SQL Server 2014.

Amazon RDS SQL Server also supports four different editions of SQL Server: Express Edition, Web Edition, Standard Edition, and Enterprise Edition. Table 7.4 highlights the relative performance, availability, and encryption differences among these editions.

TABLE 7.4 Amazon RDS SQL Server Editions Compared

Edition	Performance	Multi-AZ	Encryption
Express	+	No	KMS
Web	++++	No	KMS
Standard	++++	Yes	KMS
Enterprise	++++++++	Yes	KMS and TDE

Licensing

Amazon RDS Oracle and Microsoft SQL Server are commercial software products that require appropriate licenses to operate in the cloud. AWS offers two licensing models: *License Included* and *Bring Your Own License (BYOL)*.

License Included In the License Included model, the license is held by AWS and is included in the Amazon RDS instance price. For Oracle, License Included provides licensing for Standard Edition One. For SQL Server, License Included provides licensing for SQL Server Express Edition, Web Edition, and Standard Edition.

Bring Your Own License (BYOL) In the BYOL model, you provide your own license. For Oracle, you must have the appropriate Oracle Database license for the DB Instance class and Oracle Database edition you want to run. You can bring over Standard Edition One, Standard Edition, and Enterprise Edition.

For SQL Server, you provide your own license under the Microsoft License Mobility program. You can bring over Microsoft SQL Standard Edition and also Enterprise Edition. You are responsible for tracking and managing how licenses are allocated.

Amazon Aurora

Amazon Aurora offers enterprise-grade commercial database technology while offering the simplicity and cost effectiveness of an open source database. This is achieved by redesigning the internal components of MySQL to take a more service-oriented approach.

Like other Amazon RDS engines, Amazon Aurora is a fully managed service, is MySQL-compatible out of the box, and provides for increased reliability and performance over standard MySQL deployments. Amazon Aurora can deliver up to five times the performance of MySQL without requiring changes to most of your existing web applications. You can use the same code, tools, and applications that you use with your existing MySQL databases with Amazon Aurora.

When you first create an Amazon Aurora instance, you create a DB cluster. A DB cluster has one or more instances and includes a cluster volume that manages the data for those instances. An Amazon Aurora cluster volume is a virtual database storage volume that spans multiple Availability Zones, with each Availability Zone having a copy of the cluster data. An Amazon Aurora DB cluster consists of two different types of instances:

Primary Instance This is the main instance, which supports both read and write workloads. When you modify your data, you are modifying the primary instance. Each Amazon Aurora DB cluster has one primary instance.

Amazon Aurora Replica This is a secondary instance that supports only read operations. Each DB cluster can have up to 15 Amazon Aurora Replicas in addition to the primary instance. By using multiple Amazon Aurora Replicas, you can distribute the read workload among various instances, increasing performance. You can also locate your Amazon Aurora Replicas in multiple Availability Zones to increase your database availability.

Storage Options

Amazon RDS is built using Amazon Elastic Block Store (Amazon EBS) and allows you to select the right storage option based on your performance and cost requirements. Depending on the database engine and workload, you can scale up to 4 to 6TB in provisioned storage and up to 30,000 IOPS. Amazon RDS supports three storage types: Magnetic, General Purpose (Solid State Drive [SSD]), and Provisioned IOPS (SSD). Table 7.5 highlights the relative size, performance, and cost differences between types.

TABLE 7.5 Amazon RDS Storage Types

	Magnetic	General Purpose (SSD)	Provisioned IOPS (SSD)
Size	+++	+++++	+++++
Performance	+	+++	+++++
Cost	++	+++	+++++

Magnetic Magnetic storage, also called standard storage, offers cost-effective storage that is ideal for applications with light I/O requirements.

General Purpose (SSD) General purpose (SSD)-backed storage, also called gp2, can provide faster access than magnetic storage. This storage type can provide burst performance to meet spikes and is excellent for small- to medium-sized databases.

Provisioned IOPS (SSD) Provisioned IOPS (SSD) storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads, that are sensitive to storage performance and consistency in random access I/O throughput.


For most applications, General Purpose (SSD) is the best option and provides a good mix of lower-cost and higher-performance characteristics.

Backup and Recovery

Amazon RDS provides a consistent operational model for backup and recovery procedures across the different database engines. Amazon RDS provides two mechanisms for backing up the database: automated backups and manual snapshots. By using a combination of both techniques, you can design a backup recovery model to protect your application data.

Each organization typically will define a *Recovery Point Objective (RPO)* and *Recovery Time Objective (RTO)* for important applications based on the criticality of the application and the expectations of the users. It's common for enterprise systems to have an RPO measured in minutes and an RTO measured in hours or even days, while some critical applications may have much lower tolerances.

RPO is defined as the maximum period of data loss that is acceptable in the event of a failure or incident. For example, many systems back up transaction logs every 15 minutes to allow them to minimize data loss in the event of an accidental deletion or hardware failure.

RTO is defined as the maximum amount of downtime that is permitted to recover from backup and to resume processing. For large databases in particular, it can take hours to restore from a full backup. In the event of a hardware failure, you can reduce your RTO to minutes by failing over to a secondary node. You should create a recovery plan that, at a minimum, lets you recover from a recent backup.

Automated Backups

An *automated backup* is an Amazon RDS feature that continuously tracks changes and backs up your database. Amazon RDS creates a storage volume snapshot of your DB Instance, backing up the entire DB Instance and not just individual databases. You can set the backup retention period when you create a DB Instance. One day of backups will be retained by default, but you can modify the retention period up to a maximum of 35 days. Keep in mind that when you delete a DB Instance, all automated backup snapshots are deleted and cannot be recovered. Manual snapshots, however, are not deleted.

Automated backups will occur daily during a configurable 30-minute maintenance window called the backup window. Automated backups are kept for a configurable number of days, called the *backup retention period*. You can restore your DB Instance to any specific time during this retention period, creating a new DB Instance.

Manual DB Snapshots

In addition to automated backups, you can perform manual *DB snapshots* at any time. A DB snapshot is initiated by you and can be created as frequently as you want. You can then restore the DB Instance to the specific state in the DB snapshot at any time. DB snapshots can be created with the Amazon RDS console or the `CreateDBSnapshot` action. Unlike automated snapshots that are deleted after the retention period, manual DB snapshots are kept until you explicitly delete them with the Amazon RDS console or the `DeleteDBSnapshot` action.

For busy databases, use Multi-AZ to minimize the performance impact of a snapshot. During the backup window, storage I/O may be suspended while your data is being backed up, and you may experience elevated latency. This I/O suspension typically lasts for the duration of the snapshot. This period of I/O suspension is shorter for Multi-AZ DB deployments because the backup is taken from the standby, but latency can occur during the backup process.

Recovery

Amazon RDS allows you to recover your database quickly whether you are performing automated backups or manual DB snapshots. You cannot restore from a DB snapshot to an existing DB Instance; a new DB Instance is created when you restore. When you restore a DB Instance, only the default DB parameter and security groups are associated with the restored

instance. As soon as the restore is complete, you should associate any custom DB parameter or security groups used by the instance from which you restored. When using automated backups, Amazon RDS combines the daily backups performed during your predefined maintenance window in conjunction with transaction logs to enable you to restore your DB Instance to any point during your retention period, typically up to the last five minutes.

High Availability with Multi-AZ

One of the most powerful features of Amazon RDS is Multi-AZ deployments, which allows you to create a database cluster across multiple Availability Zones. Setting up a relational database to run in a highly available and fault-tolerant fashion is a challenging task. With Amazon RDS Multi-AZ, you can reduce the complexity involved with this common administrative task; with a single option, Amazon RDS can increase the availability of your database using replication. Multi-AZ lets you meet the most demanding RPO and RTO targets by using synchronous replication to minimize RPO and fast failover to minimize RTO to minutes.

Multi-AZ allows you to place a secondary copy of your database in another Availability Zone for disaster recovery purposes. Multi-AZ deployments are available for all types of Amazon RDS database engines. When you create a Multi-AZ DB Instance, a primary instance is created in one Availability Zone and a secondary instance is created in another Availability Zone. You are assigned a database instance endpoint such as the following:

```
my_app_db.ch6fe7ykq1zd.us-west-2.rds.amazonaws.com
```

This endpoint is a Domain Name System (DNS) name that AWS takes responsibility for resolving to a specific IP address. You use this DNS name when creating the connection to your database. Figure 7.1 illustrates a typical Multi-AZ deployment spanning two Availability Zones.

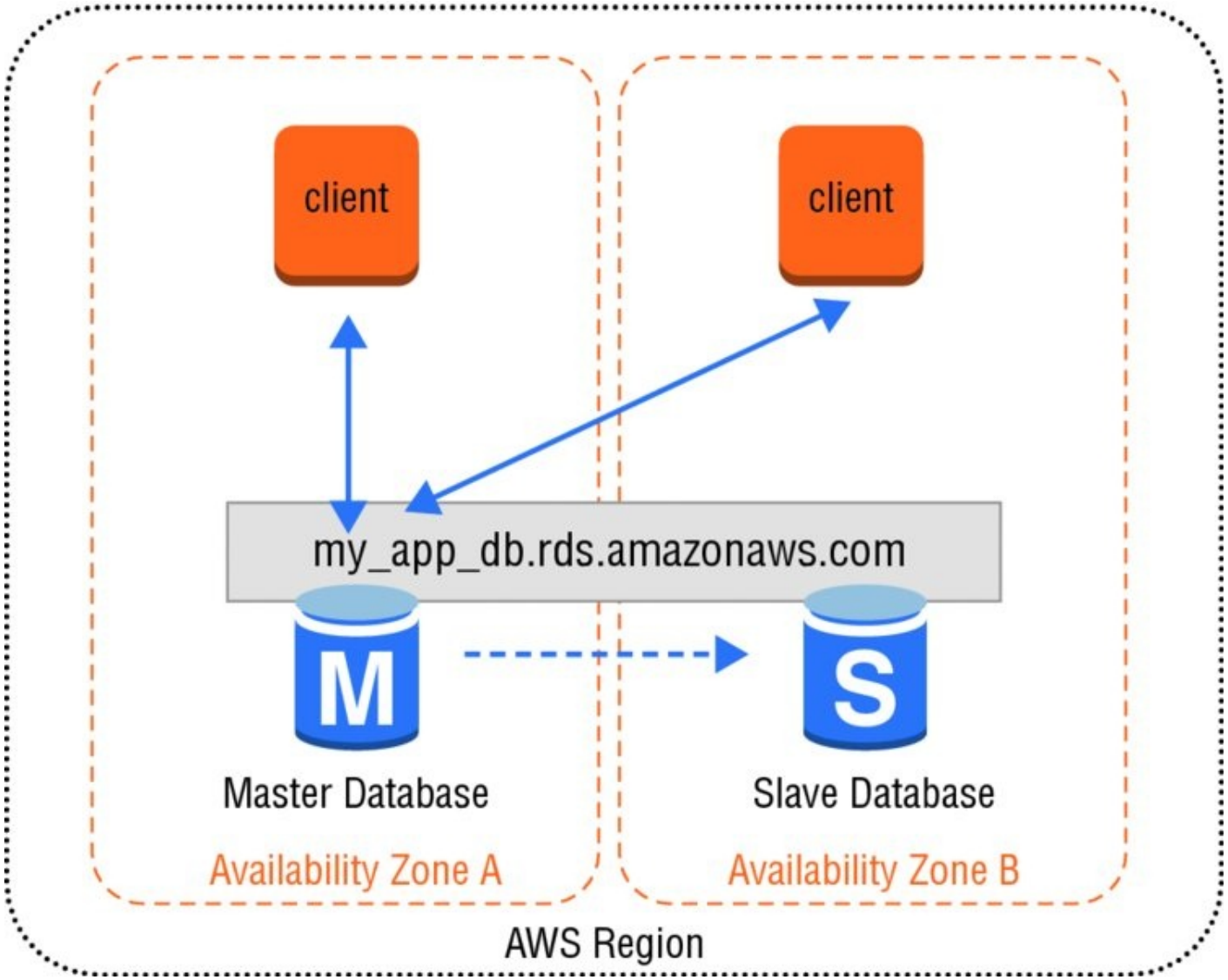


FIGURE 7.1 Multi-AZ Amazon RDS architecture

Amazon RDS automatically replicates the data from the master database or primary instance to the slave database or secondary instance using synchronous replication. Each Availability Zone runs on its own physically distinct, independent infrastructure and is engineered to be highly reliable. Amazon RDS detects and automatically recovers from the most common failure scenarios for Multi-AZ deployments so that you can resume database operations as quickly as possible without administrative intervention. Amazon RDS automatically performs a failover in the event of any of the following:

- Loss of availability in primary Availability Zone
- Loss of network connectivity to primary database
- Compute unit failure on primary database
- Storage failure on primary database

Amazon RDS will automatically fail over to the standby instance without user intervention. The DNS name remains the same, but the Amazon RDS service changes the CNAME to point to the standby. The primary DB Instance switches over automatically to the standby replica if there was an Availability Zone service disruption, if the primary DB Instance fails, or if the instance type is changed. You can also perform a manual failover of the DB Instance. Failover

between the primary and the secondary instance is fast, and the time automatic failover takes to complete is typically one to two minutes.

 It is important to remember that Multi-AZ deployments are for disaster recovery only; they are not meant to enhance database performance. The standby DB Instance is not available to offline queries from the primary master DB Instance. To improve database performance using multiple DB Instances, use read replicas or other DB caching technologies such as Amazon ElastiCache.

Scaling Up and Out

As the number of transactions increase to a relational database, scaling up, or vertically, by getting a larger machine allows you to process more reads and writes. Scaling out, or horizontally, is also possible, but it is often more difficult. Amazon RDS allows you to scale compute and storage vertically, and for some DB engines, you can scale horizontally.

Vertical Scalability

Adding additional compute, memory, or storage resources to your database allows you to process more transactions, run more queries, and store more data. Amazon RDS makes it easy to scale up or down your database tier to meet the demands of your application. Changes can be scheduled to occur during the next maintenance window or to begin immediately using the `ModifyDBInstance` action.

To change the amount of compute and memory, you can select a different DB Instance class of the database. After you select a larger or smaller DB Instance class, Amazon RDS automates the migration process to a new class with only a short disruption and minimal effort.

You can also increase the amount of storage, the storage class, and the storage performance for an Amazon RDS Instance. Each database instance can scale from 5GB up to 6TB in provisioned storage depending on the storage type and engine. Storage for Amazon RDS can be increased over time as needs grow with minimal impact to the running database. Storage expansion is supported for all of the database engines except for SQL Server.

Horizontal Scalability with Partitioning

A relational database can be scaled vertically only so much before you reach the maximum instance size. Partitioning a large relational database into multiple instances or shards is a common technique for handling more requests beyond the capabilities of a single instance.

Partitioning, or *sharding*, allows you to scale horizontally to handle more users and requests but requires additional logic in the application layer. The application needs to decide how to route database requests to the correct shard and becomes limited in the types of queries that can be performed across server boundaries. NoSQL databases like Amazon DynamoDB or Cassandra are designed to scale horizontally.

Horizontal Scalability with Read Replicas

Another important scaling technique is to use *read replicas* to offload read transactions from the primary database and increase the overall number of transactions. Amazon RDS supports read replicas that allow you to scale out elastically beyond the capacity constraints of a single DB Instance for read-heavy database workloads.

There are a variety of use cases where deploying one or more read replica DB Instances is helpful. Some common scenarios include:

- Scale beyond the capacity of a single DB Instance for read-heavy workloads.
- Handle read traffic while the source DB Instance is unavailable. For example, due to I/O suspension for backups or scheduled maintenance, you can direct read traffic to a replica.
- Offload reporting or data warehousing scenarios against a replica instead of the primary DB Instance.

For example, a blogging website may have very little write activity except for the occasional comment, and the vast majority of database activity will be read-only. By offloading some or all of the read activity to one or more read replicas, the primary database instance can focus on handling the writes and replicating the data out to the replicas.

Read replicas are currently supported in Amazon RDS for MySQL, PostgreSQL, MariaDB, and Amazon Aurora. Amazon RDS uses the MySQL, MariaDB, and PostgreSQL DB engines' built-in replication functionality to create a special type of DB Instance, called a read replica, from a source DB Instance. Updates made to the source DB Instance are asynchronously copied to the read replica. You can reduce the load on your source DB Instance by routing read queries from your applications to the read replica.


You can create one or more replicas of a database within a single AWS Region or across multiple AWS Regions. To enhance your disaster recovery capabilities or reduce global latencies, you can use cross-region read replicas to serve read traffic from a region closest to your global users or migrate your databases across AWS Regions.

Security

Securing your Amazon RDS DB Instances and relational databases requires a comprehensive plan that addresses the many layers commonly found in database-driven systems. This includes the infrastructure resources, the database, and the network.

Protect access to your infrastructure resources using AWS Identity and Access Management (IAM) policies that limit which actions AWS administrators can perform. For example, some key administrator actions that can be controlled in IAM include `CreateDBInstance` and `DeleteDBInstance`.

Another security best practice is to deploy your Amazon RDS DB Instances into a private subnet within an Amazon Virtual Private Cloud (Amazon VPC) that limits network access to the DB Instance. Before you can deploy into an Amazon VPC, you must first create a *DB subnet group* that predefines which subnets are available for Amazon RDS deployments. Further, restrict network access using network Access Control Lists (ACLs) and security groups to limit inbound traffic to a short list of source IP addresses.

At the database level, you will also need to create users and grant them permissions to read and write to your databases. Access to the database is controlled using the database engine-specific access control and user management mechanisms. Create users at the database level with strong passwords that you rotate frequently.

Finally, protect the confidentiality of your data in transit and at rest with multiple encryption capabilities provided with Amazon RDS. Security features vary slightly from one engine to another, but all engines support some form of in-transit encryption and also at-rest encryption. You can securely connect a client to a running DB Instance using Secure Sockets Layer (SSL) to protect data in transit. Encryption at rest is possible for all engines using the Amazon Key Management Service (KMS) or *Transparent Data Encryption (TDE)*. All logs, backups, and snapshots are encrypted for an encrypted Amazon RDS instance.

Summary

In this chapter, you learned the basic concepts of relational databases, data warehouses, and NoSQL databases. You also learned about the benefits and features of AWS managed database services Amazon RDS, Amazon Redshift, and Amazon DynamoDB.

Amazon RDS manages the heavy lifting involved in administering a database infrastructure and software and lets you focus on building the relational schemas that best fit your use case and the performance tuning to optimize your queries.

Amazon RDS supports popular open-source and commercial database engines and provides a consistent operational model for common administrative tasks. Increase your availability by running a master-slave configuration across Availability Zones using Multi-AZ deployment. Scale your application and increase your database read performance using read replicas.

Amazon Redshift allows you to deploy a data warehouse cluster that is optimized for analytics and reporting workloads within minutes. Amazon Redshift distributes your records using columnar storage and parallelizes your query execution across multiple compute nodes to deliver fast query performance. Amazon Redshift clusters can be scaled up or down to support large, petabyte-scale databases using SSD or magnetic disk storage.

Connect to Amazon Redshift clusters using standard SQL clients with JDBC/ODBC drivers and execute SQL queries using many of the same analytics and ETL tools that you use today. Load data into your Amazon Redshift clusters using the `COPY` command to bulk import flat files stored in Amazon S3, then run standard `SELECT` commands to search and query the table.

Back up both your Amazon RDS databases and Amazon Redshift clusters using automated and manual snapshots to allow for point-in-time recovery. Secure your Amazon RDS and Amazon Redshift databases using a combination of IAM, database-level access control, network-level access control, and data encryption techniques.

Amazon DynamoDB simplifies the administration and operations of a NoSQL database in the cloud. Amazon DynamoDB allows you to create tables quickly that can scale to an unlimited number of items and configure very high levels of provisioned read and write capacity.

Amazon DynamoDB tables provide a flexible data storage mechanism that only requires a primary key and allows for one or more attributes. Amazon DynamoDB supports both simple scalar data types like String and Number, and also more complex structures using List and Map. Secure your Amazon DynamoDB tables using IAM and restrict access to items and attributes using fine-grained access control.

Amazon DynamoDB will handle the difficult task of cluster and partition management and provide you with a highly available database table that replicates data across Availability Zones for increased durability. Track and process recent changes by tapping into Amazon DynamoDB Streams.

Exam Essentials

Know what a relational database is. A relational database consists of one or more tables. Communication to and from relational databases usually involves simple SQL queries, such as "Add a new record," or "What is the cost of product *x*?" These simple queries are often referred to as OLTP.

Understand which databases are supported by Amazon RDS. Amazon RDS currently supports six relational database engines:

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

Understand the operational benefits of using Amazon RDS. Amazon RDS is a managed service provided by AWS. AWS is responsible for patching, antivirus, and management of the underlying guest OS for Amazon RDS. Amazon RDS greatly simplifies the process of setting a secondary slave with replication for failover and setting up read replicas to offload queries.

Remember that you cannot access the underlying OS for Amazon RDS DB instances. You cannot use Remote Desktop Protocol (RDP) or SSH to connect to the underlying OS. If you need to access the OS, install custom software or agents, or want to use a database engine not supported by Amazon RDS, consider running your database on Amazon EC2 instead.

Know that you can increase availability using Amazon RDS Multi-AZ deployment. Add fault tolerance to your Amazon RDS database using Multi-AZ deployment. You can quickly set up a secondary DB Instance in another Availability Zone with Multi-AZ for rapid failover.

Understand the importance of RPO and RTO. Each application should set RPO and RTO targets to define the amount of acceptable data loss and also the amount of time required to recover from an incident. Amazon RDS can be used to meet a wide range of RPO and RTO requirements.

Understand that Amazon RDS handles Multi-AZ failover for you. If your primary Amazon RDS Instance becomes unavailable, AWS fails over to your secondary instance in another Availability Zone automatically. This failover is done by pointing your existing database endpoint to a new IP address. You do not have to change the connection string manually; AWS handles the DNS change automatically.

Remember that Amazon RDS read replicas are used for scaling out and increased performance. This replication feature makes it easy to scale out your read-intensive databases. Read replicas are currently supported in Amazon RDS for MySQL, PostgreSQL,

and Amazon Aurora. You can create one or more replicas of a database within a single AWS Region or across multiple AWS Regions. Amazon RDS uses native replication to propagate changes made to a source DB Instance to any associated read replicas. Amazon RDS also supports cross-region read replicas to replicate changes asynchronously to another geography or AWS Region.

Know what a NoSQL database is. NoSQL databases are non-relational databases, meaning that you do not have to have an existing table created in which to store your data. NoSQL databases come in the following formats:

- Document databases
- Graph stores
- Key/value stores
- Wide-column stores

Remember that Amazon DynamoDB is AWS NoSQL service. You should remember that for NoSQL databases, AWS provides a fully managed service called Amazon DynamoDB. Amazon DynamoDB is an extremely fast NoSQL database with predictable performance and high scalability. You can use Amazon DynamoDB to create a table that can store and retrieve any amount of data and serve any level of request traffic. Amazon DynamoDB automatically spreads the data and traffic for the table over a sufficient number of partitions to handle the request capacity specified by the customer and the amount of data stored, while maintaining consistent and fast performance.

Know what a data warehouse is. A data warehouse is a central repository for data that can come from one or more sources. This data repository would be used for query and analysis using OLAP. An organization's management typically uses a data warehouse to compile reports on specific data. Data warehouses are usually queried with highly complex queries.

Remember that Amazon Redshift is AWS data warehouse service. You should remember that Amazon Redshift is Amazon's data warehouse service. Amazon Redshift organizes the data by column instead of storing data as a series of rows. Because only the columns involved in the queries are processed and columnar data is stored sequentially on the storage media, column-based systems require far fewer I/Os, which greatly improves query performance. Another advantage of columnar data storage is the increased compression, which can further reduce overall I/O.

the AWS Security Token Service. Use the temporary security credentials to sign your requests to Amazon DynamoDB. Amazon DynamoDB is accessible via SSL-encrypted endpoints, and the encrypted endpoints are accessible from both the Internet and from within Amazon EC2.

Amazon Relational Database Service (Amazon RDS) Security

Amazon Relational Database Service (Amazon RDS) allows you to quickly create a relational Database Instance (DB Instance) and flexibly scale the associated compute resources and storage capacity to meet application demand. Amazon RDS manages the database instance on your behalf by performing backups, handling failover, and maintaining the database software. As of the time of this writing, Amazon RDS is available for MySQL, Oracle, Microsoft SQL Server, MariaDB, Amazon Aurora, and PostgreSQL database engines.

Amazon RDS has multiple features that enhance reliability for critical production databases, including DB security groups, permissions, SSL connections, automated backups, DB snapshots, and multiple Availability Zone (Multi-AZ) deployments. DB Instances can also be deployed in an Amazon VPC for additional network isolation.

Access Control When you first create a DB Instance within Amazon RDS, you will create a master user account, which is used only within the context of Amazon RDS to control access to your DB Instance(s). The master user account is a native database user account that allows you to log on to your DB Instance with all database privileges. You can specify the master user name and password you want associated with each DB Instance when you create the DB Instance. After you have created your DB Instance, you can connect to the database using the master user credentials. Subsequently, you can create additional user accounts so that you can restrict who can access your DB Instance.

You can control Amazon RDS DB Instance access via *DB security groups*, which are similar to Amazon EC2 security groups but not interchangeable. DB security groups act like a firewall controlling network access to your DB Instance. DB security groups default to deny all access mode, and customers must specifically authorize network ingress. There are two ways of doing this:

- Authorizing a network IP range
- Authorizing an existing Amazon EC2 security group

DB security groups only allow access to the database server port (all others are blocked) and can be updated without restarting the Amazon RDS DB Instance, which gives you seamless control of their database access.

Using AWS IAM, you can further control access to your Amazon RDS DB instances. AWS IAM enables you to control what Amazon RDS operations each individual AWS IAM user has permission to call.

Network Isolation For additional network access control, you can run your DB Instances in an Amazon VPC. Amazon VPC enables you to isolate your DB Instances by specifying the IP range you want to use and connect to your existing IT infrastructure through industry-standard encrypted IPsec VPN. Running Amazon RDS in a VPC enables you to have a DB instance within a private subnet. You can also set up a virtual private gateway that extends your corporate network into your VPC, and allows access to the RDS DB instance in that VPC.

For Multi-AZ deployments, defining a subnet for all Availability Zones in a region, will allow

Amazon RDS to create a new standby in another Availability Zone should the need arise. You can create DB subnet groups, which are collections of subnets that you may want to designate for your Amazon RDS DB Instances in an Amazon VPC. Each DB subnet group should have at least one subnet for every Availability Zone in a given region. In this case, when you create a DB Instance in an Amazon VPC, you select a DB subnet group; Amazon RDS then uses that DB subnet group and your preferred Availability Zone to select a subnet and an IP address within that subnet. Amazon RDS creates and associates an Elastic Network Interface to your DB Instance with that IP address.

DB Instances deployed within an Amazon VPC can be accessed from the Internet or from Amazon EC2 instances outside the Amazon VPC via VPN or bastion hosts that you can launch in your public subnet. To use a bastion host, you will need to set up a public subnet with an Amazon EC2 instance that acts as a SSH Bastion. This public subnet must have an Internet gateway and routing rules that allow traffic to be directed via the SSH host, which must then forward requests to the private IP address of your Amazon RDS DB Instance.

DB security groups can be used to help secure DB Instances within an Amazon VPC. In addition, network traffic entering and exiting each subnet can be allowed or denied via network ACLs. All network traffic entering or exiting your Amazon VPC via your IPsec VPN connection can be inspected by your on-premises security infrastructure, including network firewalls and intrusion detection systems.

Encryption You can encrypt connections between your application and your DB Instance using SSL. For MySQL and SQL Server, Amazon RDS creates an SSL certificate and installs the certificate on the DB Instance when the instance is provisioned. For MySQL, you launch the MySQL client using the `--ssl_ca` parameter to reference the public key in order to encrypt connections. For SQL Server, download the public key and import the certificate into your Windows operating system. Oracle RDS uses Oracle native network encryption with a DB Instance. You simply add the native network encryption option to an option group and associate that option group with the DB Instance. After an encrypted connection is established, data transferred between the DB Instance and your application will be encrypted during transfer. You can also require your DB Instance to accept only encrypted connections.

Amazon RDS supports Transparent Data Encryption (TDE) for SQL Server (SQL Server Enterprise Edition) and Oracle (part of the Oracle Advanced Security option available in Oracle Enterprise Edition). The TDE feature automatically encrypts data before it is written to storage and automatically decrypts data when it is read from storage. If you require your MySQL data to be encrypted while at rest in the database, your application must manage the encryption and decryption of data.

Note that SSL support within Amazon RDS is for encrypting the connection between your application and your DB Instance; it should not be relied on for authenticating the DB Instance itself. While SSL offers security benefits, be aware that SSL encryption is a compute intensive operation and will increase the latency of your database connection.

Automated Backups and DB Snapshots Amazon RDS provides two different methods for backing up and restoring your DB Instance(s): automated backups and Database Snapshots (DB Snapshots). Turned on by default, the automated backup feature of Amazon RDS enables point-in-time recovery for your DB Instance. Amazon RDS will back up your database and transaction logs and store both for a user-specified retention period. This allows

you to restore your DB Instance to any second during your retention period, up to the last five minutes. Your automatic backup retention period can be configured to up to 35 days.

DB Snapshots are user-initiated backups of your DB Instance. These full database backups are stored by Amazon RDS until you explicitly delete them. You can copy DB snapshots of any size and move them between any of AWS public regions, or copy the same snapshot to multiple regions simultaneously. You can then create a new DB Instance from a DB Snapshot whenever you desire.

During the backup window, storage I/O may be suspended while your data is being backed up. This I/O suspension typically lasts a few minutes. This I/O suspension is avoided with Multi-AZ DB deployments, because the backup is taken from the standby.

DB Instance Replication AWS Cloud computing resources are housed in highly available data center facilities in different regions of the world, and each region contains multiple distinct locations called Availability Zones. Each Availability Zone is engineered to be isolated from failures in other Availability Zones and provide inexpensive, low-latency network connectivity to other Availability Zones in the same region.

To architect for high availability of your Oracle, PostgreSQL, or MySQL databases, you can run your Amazon RDS DB Instance in several Availability Zones, an option called a *Multi-AZ deployment*. When you select this option, AWS automatically provisions and maintains a synchronous standby replica of your DB Instance in a different Availability Zone. The primary DB Instance is synchronously replicated across Availability Zones to the standby replica. In the event of DB Instance or Availability Zone failure, Amazon RDS will automatically failover to the standby so that database operations can resume quickly without administrative intervention.

For customers who use MySQL and need to scale beyond the capacity constraints of a single DB Instance for read-heavy database workloads, Amazon RDS provides a read replica option. After you create a read replica, database updates on the source DB Instance are replicated to the read replica using MySQL's native, asynchronous replication. You can create multiple read replicas for a given source DB instance and distribute your application's read traffic among them. Read replicas can be created with Multi-AZ deployments to gain read scaling benefits in addition to the enhanced database write availability and data durability provided by Multi-AZ deployments.

Automatic Software Patching Amazon RDS will make sure that the relational database software powering your deployment stays up-to-date with the latest patches. When necessary, patches are applied during a maintenance window that you can control. You can think of the Amazon RDS maintenance window as an opportunity to control when DB Instance modifications (such as scaling DB Instance class) and software patching occur, in the event either are requested or required. If a maintenance event is scheduled for a given week, it will be initiated and completed at some point during the 30-minute maintenance window you identify.

The only maintenance events that require Amazon RDS to take your DB Instance offline are scale compute operations (which generally take only a few minutes from start to finish) or required software patching. Required patching is automatically scheduled only for patches that are related to security and durability. Such patching occurs infrequently (typically once every few months) and should seldom require more than a fraction of your maintenance

window. If you do not specify a preferred weekly maintenance window when creating your DB Instance, a 30-minute default value is assigned. If you want to modify when maintenance is performed on your behalf, you can do so by modifying your DB Instance in the AWS Management Console or by using the `ModifyDBInstance` API. Each of your DB Instances can have different preferred maintenance windows, if you so choose.

Running your DB Instance in a Multi-AZ deployment can further reduce the impact of a maintenance event, as Amazon RDS will conduct maintenance via the following steps:

1. Perform maintenance on standby.
2. Promote standby to primary.
3. Perform maintenance on old primary, which becomes the new standby.

When an Amazon RDS DB Instance deletion API (`DeleteDBInstance`) is run, the DB Instance is marked for deletion. After the instance no longer indicates deleting status, it has been removed. At this point, the instance is no longer accessible, and unless a final snapshot copy was asked for, it cannot be restored and will not be listed by any of the tools or APIs.

Amazon Redshift Security

Amazon Redshift is a petabyte-scale SQL data warehouse service that runs on highly optimized and managed AWS compute and storage resources. The service has been architected not only to scale up or down rapidly, but also to improve query speeds significantly even on extremely large datasets. To increase performance, Amazon Redshift uses techniques such as columnar storage, data compression, and zone maps to reduce the amount of I/O needed to perform queries. It also has a Massively Parallel Processing (MPP) architecture, parallelizing and distributing SQL operations to take advantage of all available resources.

Cluster Access By default, clusters that you create are closed to everyone. Amazon Redshift enables you to configure firewall rules (security groups) to control network access to your data warehouse cluster. You can also run Amazon Redshift inside an Amazon VPC to isolate your data warehouse cluster in your own virtual network and connect it to your existing IT infrastructure using industry-standard encrypted IPsec VPN.

The AWS account that creates the cluster has full access to the cluster. Within your AWS account, you can use AWS IAM to create user accounts and manage permissions for those accounts. By using IAM, you can grant different users permission to perform only the cluster operations that are necessary for their work. Like all databases, you must grant permission in Amazon Redshift at the database level in addition to granting access at the resource level. Database users are named user accounts that can connect to a database and are authenticated when they log in to Amazon Redshift. In Amazon Redshift, you grant database user permissions on a per-cluster basis instead of on a per-table basis. However, users can see data only in the table rows that were generated by their own activities; rows generated by other users are not visible to them.

The user who creates a database object is its owner. By default, only a super user or the owner of an object can query, modify, or grant permissions on the object. For users to use an object, you must grant the necessary permissions to the user or the group that contains the user. In addition, only the owner of an object can modify or delete it.

Amazon Simple Storage Service (Amazon S3) Amazon S3 allows you to upload and retrieve data at any time, from anywhere on the web. Access to data stored in Amazon S3 is restricted by default; only bucket and object owners have access to the Amazon S3 resources they create. You can securely upload and download data to Amazon S3 via the SSL-encrypted endpoints. Amazon S3 supports several methods to encrypt data at rest.

Amazon Glacier Amazon Glacier service provides low-cost, secure, and durable storage. You can securely upload and download data to Amazon Glacier via the SSL-encrypted endpoints, and the service automatically encrypts the data using AES-256 and stores it durably in an immutable form.

AWS Storage Gateway AWS Storage Gateway service connects your on-premises software appliance with cloud-based storage to provide seamless and secure integration between your IT environment and AWS storage infrastructure. Data is asynchronously transferred from your on-premises storage hardware to AWS over SSL and stored encrypted in Amazon S3 using AES-256.

Database

Amazon DynamoDB Amazon DynamoDB is a managed NoSQL database service that provides fast and predictable performance with seamless scalability. You can control access at the database level by creating database-level permissions that allow or deny access to items (rows) and attributes (columns) based on the needs of your application.

Amazon Relational Database Service (RDS) Amazon RDS allows you to quickly create a relational DB Instance and flexibly scale the associated compute resources and storage capacity to meet application demand. You can control Amazon RDS DB Instance access via DB security groups, which act like a firewall controlling network access to your DB Instance. Database security groups default to deny all access mode, and customers must specifically authorize network ingress. Amazon RDS is supported within an Amazon VPC, and for Multi-AZ deployments, defining a subnet for all Availability Zones in a region will allow Amazon RDS to create a new standby in another Availability Zone should the need arise. You can encrypt connections between your application and your DB Instance using SSL, and you can encrypt data at rest within Amazon RDS instances for all database engines.

Amazon Redshift Amazon Redshift is a petabyte-scale SQL data warehouse service that runs on highly optimized and managed AWS compute and storage resources. The service enables you to configure firewall rules (security groups) to control network access to your data warehouse cluster. Database users are named user accounts that can connect to a database and are authenticated when they log in to Amazon Redshift. In Amazon Redshift, you grant database user permissions on a per-cluster basis instead of on a per-table basis. You may choose for Amazon Redshift to store all data in user-created tables in an encrypted format using hardware-accelerated AES-256 block encryption keys. This includes all data written to disk and also any backups. Amazon Redshift uses a four-tier, key-based architecture for encryption. These keys consist of data encryption keys, a database key, a cluster key, and a master key.

Amazon ElastiCache Amazon ElastiCache is a web service that makes it easy to set up, manage, and scale distributed in-memory cache environments in the cloud. Amazon ElastiCache allows you to control access to your Cache Clusters using Cache Security Groups.