aws

Getting Started Guide

Amazon Relational Database Service

Amazon Relational Database Service: Getting Started Guide

Table of Contents

Getting started with Amazon Relational Database Service

Amazon Relational Database Service (Amazon RDS) is a managed database service that simplifies the process of setting up, operating, and scaling relational databases in the cloud. It supports popular database engines such as MySQL, PostgreSQL, MariaDB, Oracle, and SQL Server. With Amazon RDS, you can focus on your applications while AWS handles time-consuming database tasks like backups, software patching, monitoring, and hardware provisioning.

This getting started guide helps you navigate the initial setup and use of Amazon RDS. It provides step-by-step instructions and examples to guide you through creating, securing, and connecting to your first DB instance. Whether you're new to cloud databases or transitioning from on-premises solutions, this guide ensures you can get started with confidence.

> **Note**
>
> This guide focuses primarily on the core tasks required to get started with Amazon RDS. For comprehensive documentation of Amazon RDS, including advanced features and configurations, see the Amazon RDS User Guide.

Topics

Purpose and benefits of Amazon RDS

Amazon RDS removes the operational complexities of database management, so you can focus on developing and running your applications. By automating essential tasks like backups, scaling, and high availability, you can save time and reduce administrative overhead.

Key benefits for new users include:

- **Ease of setup** – Launch a DB instance using guided workflows or API calls.

- **Scalability** – Adjust compute and storage resources to meet changing demands.
- **Reliability** – Built-in fault tolerance and automated backups ensure your data is safe and available.
- **Security** – RDS supports encryption, network isolation, and identity-based access control to protect your data.

As you explore this guide, you'll learn how to create, secure, and connect to your first DB instance, which provides the foundation for effective database management in the cloud.

Key concepts and architecture of Amazon RDS

Amazon RDS is a fully managed service that simplifies the deployment and management of relational databases in the cloud. It supports multiple database engines and automates tasks like provisioning, patching, backups, and scaling so you can focus on application development.

RDS includes features such as Multi-AZ deployments, automated backups, and resource scaling. Understanding these key concepts and architectural components is important for optimizing performance, security, and availability.

Topics

Architecture

Amazon RDS is built on a distributed architecture designed to provide scalability, availability, and reliability for your database workloads.

Database instances

A database instance (DB instance) provides the underlying infrastructure for your database. It includes compute (CPU and memory), storage, and IOPS, all managed by AWS. You're responsible for your database software and configurations, while AWS handles hardware provisioning, maintenance, and backups.

For more information, see Amazon RDS DB instances.

Multi-Availability Zone deployments

For high availability, RDS can deploy a database across multiple Availability Zones (AZs). RDS replicates the primary database instance in one AZ to a standby instance in another AZ. In the event of failure, RDS automatically switches to the standby instance, which ensures minimal downtime.

For more information, see Configuring and managing a Multi-AZ deployment for Amazon RDS.

Storage

RDS provides multiple storage options, including General Purpose, Provisioned IOPS, and Magnetic storage, to meet different performance needs. Storage automatically scales as needed, and replication provides data durability.

For more information, see Amazon RDS DB instance storage.

Network integration

You can launch RDS instances within a Virtual Private Cloud (VPC), which lets you define network configurations and isolate your database in a secure environment. VPC integration also provides secure communication with other AWS resources like EC2 instances, Lambda functions, and S3 buckets.

For more information, see Amazon VPC and Amazon RDS.

Features

RDS includes a broad set of features that make database management easier and more efficient.

Automated backups and snapshots

RDS automatically takes daily backups of your DB instance, retaining backup data for a configurable retention period. You can take manual snapshots in order to preserve a point-in-time backup of your database, which you can later restore to create a new instance.

For more information, see Introduction to backups.

Scalability

RDS supports both vertical and horizontal scaling. To scale vertically, modify your instance type to handle increased database load. To scale horizontally, create read replicas to distribute read traffic across multiple instances and improve performance.

For more information, see Working with storage for Amazon RDS DB instances.

Security

RDS provides security features to safeguard your data and manage access. It supports encryption at rest and in transit using SSL and AWS Key Management Service (AWS KMS). RDS integrates with AWS Identity and Access Management (IAM) to control access to DB instances. It also supports Kerberos authentication for centralized and secure user authentication. Additionally, RDS integrates with AWS Secrets Manager to securely manage and rotate database credentials, which minimizes the risks of hard-coded secrets. You can isolate RDS instances within a Virtual Private Cloud (VPC), and use security groups and network access control lists (ACLs) for fine-grained network access control.

For more information, see Security in Amazon RDS.

Blue/Green Deployments

RDS Blue/Green Deployments help you safely deploy application updates or new configurations by running two separate environments (blue and green). This method reduces downtime and minimizes risks during the deployment process.

For more information, see Using Amazon RDS Blue/Green Deployments for database updates.

RDS Custom

With RDS Custom, you can run custom database engines on Amazon RDS. It provides full control over database configurations and extensions while benefiting from the scalability and management features of RDS. It's ideal for workloads that require more flexibility than standard RDS instances.

For more information, see Amazon RDS Custom.

RDS Proxy

RDS Proxy is a fully managed, highly available database proxy that sits between your application and the RDS database. It helps improve application scalability, availability, and security by managing database connections efficiently, reducing the load on your database and minimizing connection overhead.

For more information, see Amazon RDS Proxy.

Zero-ETL integrations

RDS supports zero-ETL integrations, which provide data movement between your RDS databases and Amazon Redshift without the need for complex extraction, transformation, and loading (ETL) processes. This simplifies data integration workflows and accelerates analytics.

For more information, see Amazon RDS zero-ETL integrations with Amazon Redshift.

Monitoring and Performance Insights

RDS provides monitoring through Amazon CloudWatch, which tracks key metrics such as CPU usage, storage space, and I/O activity. RDS Performance Insights offers deeper visibility into database performance, helping you identify bottlenecks and optimize queries.

For more information, see Monitoring metrics in an Amazon RDS instance.

Maintenance and patching

RDS manages software patching automatically within a specified maintenance window. This ensures that your DB instances stay up-to-date with the latest security patches and feature improvements without requiring manual intervention.

For more information, see Maintaining a DB instance.

Cross-Region replication

RDS supports cross-Region replication, so you can replicate your data to a different AWS Region for disaster recovery, data locality, and compliance purposes. You can use this feature to create global, low-latency applications.

For more information, see Creating a read replica in a different AWS Region.

Supported database engines

Amazon RDS supports the following database engines, so you can choose the right database technology for your application needs.

- MySQL – Popular for web applications, offering compatibility with many tools and frameworks.
- PostgreSQL – Known for advanced features, strong standards compliance, and extensibility.
- MariaDB – An open-source fork of MySQL, offering additional features and high compatibility.
- Oracle – Provides enterprise-grade capabilities, including advanced performance, security, and high availability.

- Microsoft SQL Server – Suited for organizations relying on Microsoft technologies, offering analytics and Business Intelligence (BI) integration.
- IBM Db2 – A database for enterprise applications, known for its scalability and advanced analytics.

Each engine comes with multiple versions and configurations, so you can align with your existing application requirements or take advantage of newer features.

This guide helps you select and configure a database engine during the setup process and provides links to resources for engine-specific optimization and features.

Mapping Amazon RDS personas to this guide

This guide is structured around common Amazon RDS user roles to provide targeted, actionable guidance based on your responsibilities and objectives. The following table helps you identify the sections most relevant to your needs, whether you're a data professional optimizing database performance, a developer connecting applications, or a security professional ensuring compliance.

By aligning each chapter with specific challenges and goals, this guide offers a tailored approach to getting started with Amazon RDS.

Persona	Where to start	Why start here?
Cloud implementer	Creating your first DB instance	This chapter guides you through the process of creating a DB instance, including selecting the appropria te engine and configuring instance settings. **Your goals**: • Deploy and manage RDS resources effectively. • Ensure efficient integration into development and deployment workflows.

Persona	Where to start	Why start here?
		• Master ongoing operational tasks to ensure database availability and performance.
Security professional	Securing your DB instance	This chapter focuses on setting up critical security configurations, including IAM roles, encryption, and networking best practices. **Your goals**: • Ensure regulatory compliance. • Protect data from unauthorized access. • Set up secure networking.
Developer	Connecting to your DB instance	This chapter guides developers through connecting to RDS, including using CLI commands with practical examples. **Your goals**: • Successfully connect to the database. • Troubleshoot connectivity issues. • Automate database connections with Infrastructure as Code (IaC).

Persona	Where to start	Why start here?
Data engineer or manager	Managing your DB instance and Optimizing and scaling your DB instance	These chapters provide guidance on monitoring, diagnosing, and optimizing your RDS database to ensure reliable performance and operational efficiency. **Your goals**: • Diagnose and resolve performance issues efficiently. • Monitor database health and ensure uptime. • Implement effective logging and alerting systems.

Common challenges and how this guide addresses them

When you're getting started with Amazon RDS, you might encounter challenges in three key areas.

Topics

- Security
- Connectivity
- Performance

Security

When you're attempting to secure your database, you might find it challenging to configure IAM roles, network settings, and database authentication mechanisms. Misconfigurations can expose your database to unauthorized access or make it unusable.

This guide provides instructions for setting up secure access, including configuring VPC security groups and setting up database authentication.

Connectivity

It can be complicated to establish a reliable connection to your RDS database, especially when you work with VPCs, subnets, and public or private accessibility settings. Incorrect configurations can block applications from communicating with the database.

This guide provides instructions for connecting to your database from different environments, along with troubleshooting tips to resolve common connectivity issues.

Performance

It can be complex to choose the right instance types, storage options, and performance monitoring tools. These decisions are critical to ensuring your database performs well under varying workloads.

This guide offers advice on selecting the appropriate database configurations for your workload, enabling performance monitoring, and fine-tuning settings to maintain efficiency as your application grows.

Setting up Amazon RDS

To start using Amazon RDS, you need to set up your AWS account and create a user with administrative access. This process involves signing up for an AWS account, securing the root user, and configuring IAM Identity Center. By following these steps, you can set up your AWS environment to manage Amazon RDS instances while maintaining security best practices.

Topics

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.

2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call or text message and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.
2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

Next steps

Now that you set up and configured your AWS account, you can get started by creating your first Amazon RDS DB instance.

Next step: *Creating a DB instance*

Creating an Amazon RDS DB instance

Amazon RDS simplifies database creation so you can easily launch a fully-managed database. In this section, you'll learn how to set up your first DB instance and gain an understanding of key configurations such as choosing the right database engine, selecting an instance class, and configuring storage options. Whether you prefer to use the AWS Management Console or the AWS Command Line Interface (AWS CLI), this topic provides step-by-step instructions tailored to your needs.

What you'll accomplish:

- Learn how to choose the best database engine for your use case.
- Create your first DB instance using the AWS Management Console, with visual guidance and detailed explanations.
- (Optional) Use the AWS CLI to programmatically create and manage your DB instance.

By the end of this section, you'll have a functional, fully-managed database ready to connect with your applications and workflows.

Topics

Choosing your database engine for Amazon RDS

To set up your Amazon RDS DB instance, you must first choose the appropriate database engine. Amazon RDS supports several widely-used relational database engines, each suited to specific use cases and application requirements.

If you're new to Amazon RDS or relational databases, consider starting with MySQL or PostgreSQL. Both engines are open-source, cost-effective, and widely supported by the development community. Choose **MySQL** if your application needs broad compatibility with existing tools or frameworks, or if your workload involves simple transactional operations. Choose **PostgreSQL** if your application requires advanced features such as JSON data handling, complex queries, or support for custom extensions.

> **Note**
>
> This guide primarily uses MySQL for examples, but most steps are similar for PostgreSQL. By selecting an engine that aligns with your project requirements, you can ensure better performance and scalability as your database evolves.

Amazon RDS supports the following database engines, each designed for specific use cases and performance needs.

MySQL

- **When to choose**: Best for for web-based applications, content management systems, and workloads that require simple transactional operations. Its broad compatibility with frameworks and tools makes it an ideal choice for general-purpose use cases.
- **Key features**: Read replicas, point-in-time recovery, multiple storage engines.
- **Documentation**: Amazon RDS for MySQL

PostgreSQL

- **When to choose**: Ideal for applications that require advanced features like support for JSON, geospatial data, or complex queries. It's also well-suited for data analysis and custom application development.
- **Key features**: Extensibility with custom modules, high availability, advanced indexing options.
- **Documentation**: Amazon RDS for PostgreSQL

Oracle

- **When to choose**: Designed for enterprise-grade workloads that require high performance, advanced analytics, or extensive transactional capabilities. Common in financial services and large-scale Enterprise Resource Planning (ERP) systems.
- **Key features**: Advanced security, analytics capabilities, and database partitioning.
- **Documentation**: Amazon RDS for Oracle

SQL Server

- **When to choose**: Best suited for organizations that use the Microsoft ecosystem or developing applications with tight integration to Windows-based services. Common in Business Intelligence (BI) workloads and reporting solutions.
- **Key features**: Built-in analytics, data compression, and high availability options.
- **Documentation**: Amazon RDS for Microsoft SQL Server

MariaDB

- **When to choose**: An open-source alternative to MySQL, MariaDB is ideal if you need enterprise-grade performance and features with community-driven development.
- **Key features**: Dynamic thread pooling, enhanced replication, and compatibility with MySQL.
- **Documentation**: Amazon RDS for MariaDB

IBM Db2

- **When to choose**: Preferred for mission-critical applications that require advanced analytics, data integrity, and scalability. Common in industries such as finance and healthcare.
- **Key features**: Advanced compression, scalability, and high availability.
- **Documentation**: Amazon RDS for Db2

Next steps

Now that you selected the database engine that best suits your needs, it's time to create your first Amazon RDS DB instance.

Next step: the section called “Creating your first DB instance”

Creating your first Amazon RDS DB instance

Amazon RDS simplifies the process of setting up and managing relational databases in the cloud. This walkthrough guides you through creating your first DB instance. By the end, you'll understand how to configure and manage your RDS DB instance.

Topics

- Creating your first DB instance

Creating your first DB instance

A DB instance in Amazon RDS is the foundational building block for running a managed relational database in the cloud. This section helps you set up your first RDS DB instance.

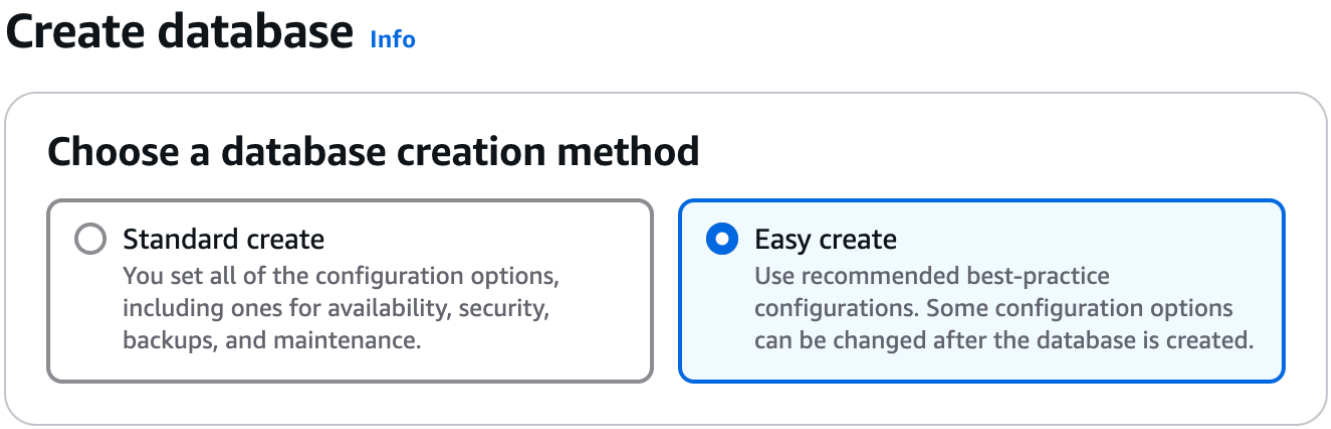
Console

This tutorial walks you through the steps to create a simple RDS for MySQL DB instance using the **Easy create** option. For comprehensive instructions to create a production DB instance, see Creating an Amazon RDS DB instance in the *Amazon RDS User Guide*.

To create a DB instance using Easy create

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Choose **Create database**.

3. For the creation method, choose **Easy create**. This method simplifies database provisioning by automatically configuring settings such as instance class, storage type, and networking settings.



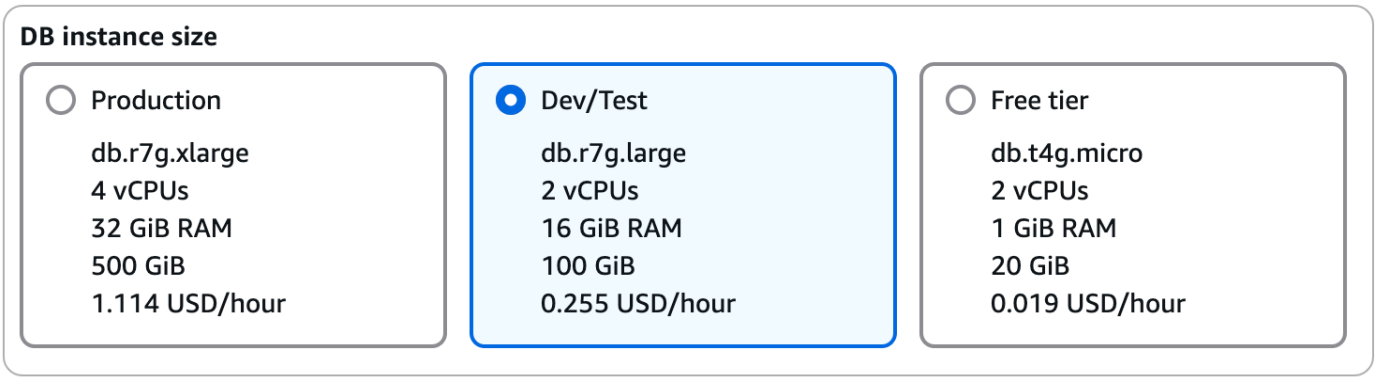
4. In this tutorial, we create a MySQL DB instance. For **Engine type**, choose **MySQL**.

   MySQL is a good starting point because it's open-source, cost-effective, and widely supported by the development community. For a full list of supported database engines, see the section called “Choosing your database engine”.

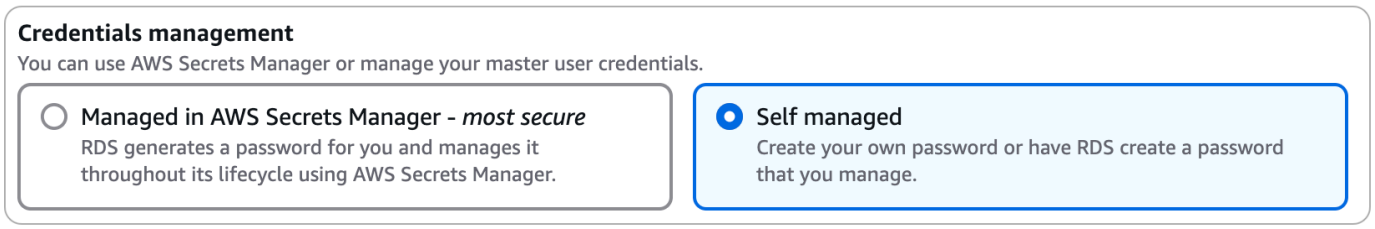
5. For **DB instance size**, choose **Free tier** or **Sandbox**. **Free tier** appears for free plan accounts. **Sandbox** appears for paid plan accounts. The **Free tier** option lets you complete the tutorial without incurring costs, so it's ideal for learning and experimentation.

   If you were an AWS Free Tier customer before July 17, 2025 and your usage exceeds the free tier limits or you select resources not covered by the free tier, you're billed at the listed hourly rate.

   The following screenshot shows the **Free tier** option.



6. (Optional) For **DB instance identifier**, enter a name for the DB instance. Alternately, keep the name that Amazon RDS generates for you.
7. For **Credentials management**, select **Self-managed**. This option lets you manage your own master user credentials.



8. For **Master password**, enter a password for the master user and confirm the password.
9. Expand **View default settings for Easy create** and review the settings that Amazon RDS automatically configures for you. You can modify some settings, such as public access and the engine version, after you create the DB instance.
10. Choose **Create database**.

    The database appears in the **Databases** list with a status of `Creating`. When the status changes to `Available`, the DB instance is ready to use.

AWS CLI

The following command creates a simple RDS for MySQL DB instance using the AWS CLI. The command replicates the **Easy create** option from the RDS console, which default settings for development and experimentation. For production-ready configurations, see Creating an Amazon RDS DB instance in the *Amazon RDS User Guide*.

First, install and configure the AWS CLI. For instructions, see Installing or updating to the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

Then, run the following command:

Example

```
aws rds create-db-instance \
  --db-instance-identifier my-db-instance \
  --db-instance-class db.t4g.micro \
  --engine mysql \
  --master-username my-username \
  --master-user-password my-password \
  --allocated-storage 20 \
  --no-publicly-accessible \
  --backup-retention-period 7 \
  --storage-type gp2 \
  --engine-version 8.0.39
```

Replace the placeholders values for the instance identifier, username, and password with your own values.

Other important settings

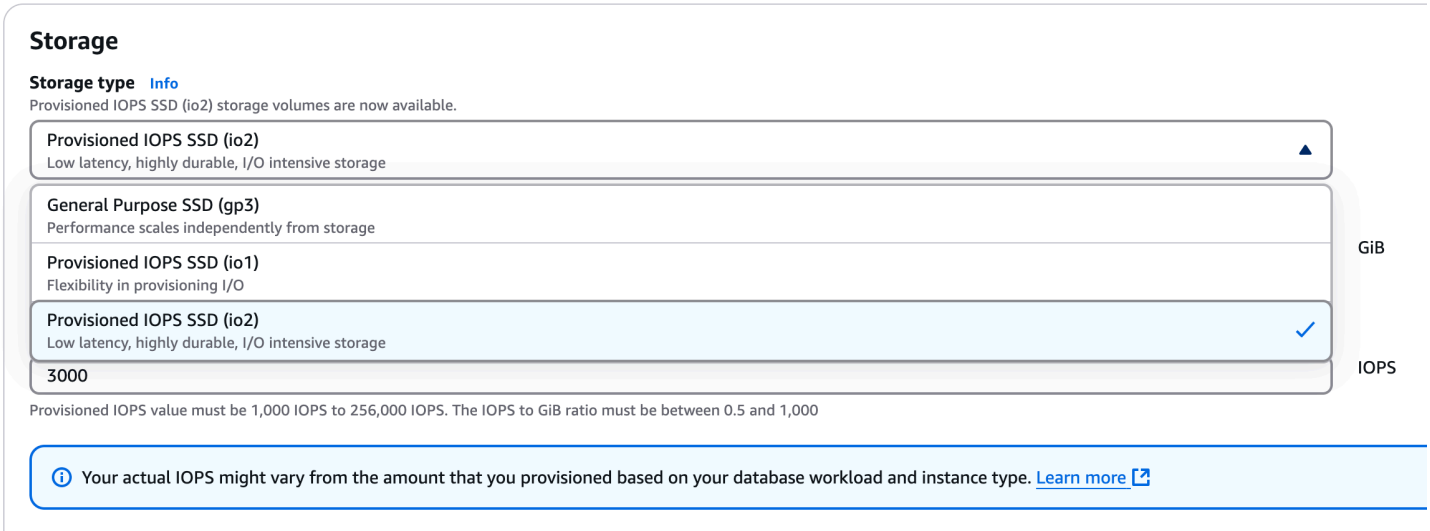
Although the **Easy create** option simplifies the process of creating a DB instance, using the **Standard create** workflow offers more control and customization over your DB instance configuration. Consider the following important settings when you create a production DB instance.

Topics

Storage allocation

In the Standard create workflow, you must specify the amount of storage for your DB instance. The type of storage, such as General Purpose SSD, Provisioned IOPS, or Magnetic, and the allocated size directly affect your database performance and cost.

- For most workloads, General Purpose SSD provides a balance between cost and performance.
- High-performance transactional applications benefit from Provisioned IOPS SSD.

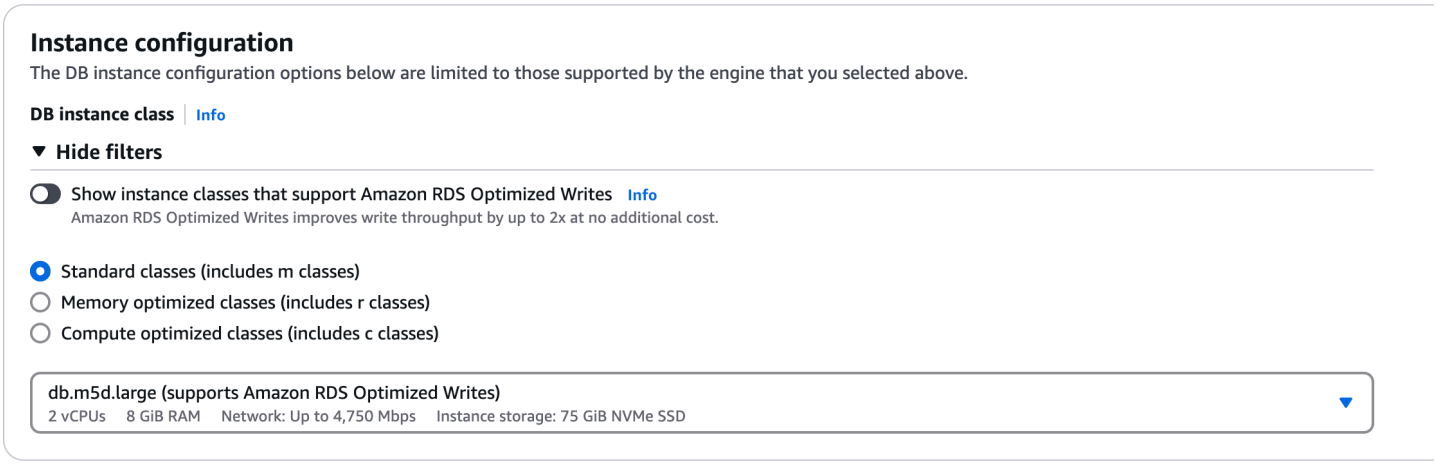


For more information, see Amazon RDS DB instance storage in the *Amazon RDS User Guide*.

Instance class

The instance class determines the allocated compute and memory capacity for your database. Selecting the right class depends on factors such as the expected workload, query volume, and application requirements.

- **Standard classes** are ideal for general-purpose workloads.
- **Memory-optimized classes** are best for applications that require high memory throughput.
- **Burstable classes** work well for applications with intermittent workloads.

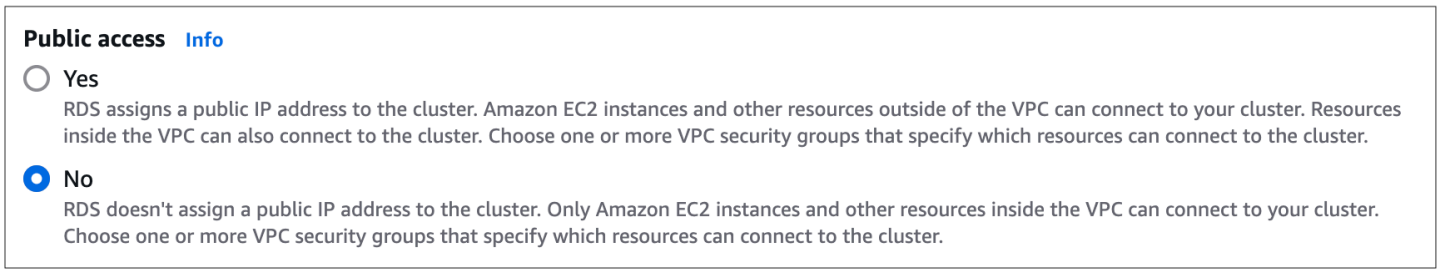


For detailed guidance, see DB instance classes in the *Amazon RDS User Guide*.

Public access

Public access controls whether users or applications can access your DB instance from outside the VPC. This setting is critical for managing connectivity and security.

- **Enable public access** to make your database accessible from external networks, such as for web applications or remote access. Configure security group rules to restrict unwanted access.
- **Disable public access** for internal applications or enhanced security, limiting connectivity to instances within your VPC.



For more information about configuring public or private access and related network settings, see Working with a DB instance in a VPC in the *Amazon RDS User Guide*.

Next steps

Now that you created your first RDS DB instance, the next step is to make sure that it's secure. Proper security configurations help protect your database and its data from unauthorized access.

In the next section, you'll learn about the following:

- Setting up public or private access.
- Configuring security groups and network settings.
- Managing database authentication and access controls.

Next step: *Securing your DB instance*

Securing your Amazon RDS DB instance

Security is a critical aspect of Amazon RDS database management, ensuring that sensitive data remains protected and accessible only to authorized users. This chapter provides an overview of key security features and configurations available in Amazon RDS, along with step-by-step guidance for setting up your environment securely.

By the end of this chapter, you will understand the following concepts:

- How to use security groups to control access to your DB instance.
- The differences between public and private DB instances and how to configure them.
- Authentication options for managing user access.

Topics

Setting up public or private access in Amazon RDS

When you create an Amazon RDS DB instance, one of the most important decisions you make is determining how others can access it. Depending on your use case, users and applications can access your database either from the internet (public access) or from a restricted private network within your AWS environment. Choosing the right access configuration is essential to balance security and connectivity.

> **Note**
>
> By default, the **Easy create** option sets your database to private access, meaning it's only accessible from within your VPC. This configuration enhances security by restricting internet exposure. However, if you need a simpler connection process for testing or development purposes, consider changing to public access. Public access allows connections from the internet.

This section helps you understand the key differences between public and private access, and the benefits and risks associated with each. By the end of this section, you'll be able to choose and set up the appropriate access type for your workload.

Topics

Public access

A publicly accessible DB instance has a public IP address, which allows users to reach it directly from the internet. Use this configuration for applications that need remote access but require strict access control. For example, you might allow access only from a corporate IP range or specific developer machines.

You can use this setup for the following use cases:

- **External applications**: You need to connect a database to applications hosted outside of AWS.
- **Testing and development**: You're developing or testing from your local machine and need direct access.

Risks: Publicly accessible databases are vulnerable to potential threats from the internet.

Considerations: If you enable public access, you must apply strict access controls. For example, you can use the following strategies:

- Configure inbound rules in the security group to allow only trusted IP addresses.
- Ensure strong authentication and encryption are in place to prevent unauthorized access.

Private access

A private DB instance is accessible only within your VPC. It doesn't have a public IP address, and you can't reach it from the internet. You can use private access for scenarios like internal microservices connecting to the database within the same VPC.

This configuration is best for the following use cases:

- **Internal applications**: Databases used by applications within the same VPC or a connected on-premises network.
- **High-security requirements**: Workloads that require stringent access control, such as financial or healthcare applications.

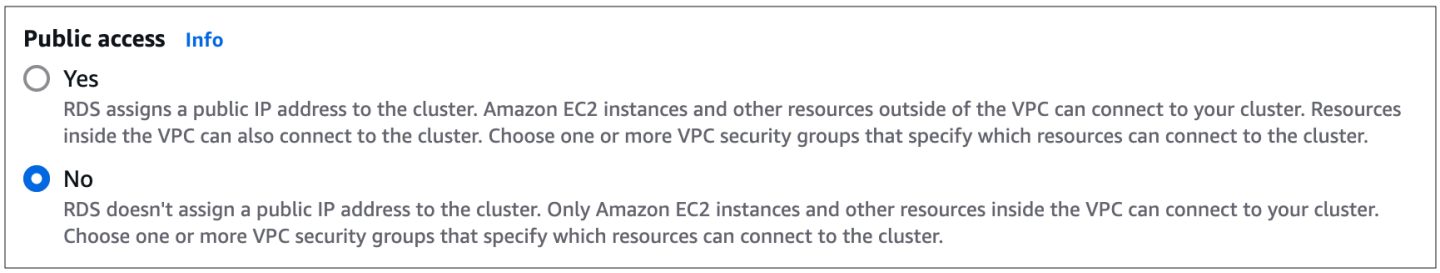
Benefits: Private access minimizes exposure to external threats because RDS isolates the database from the public internet.

Considerations: For private databases, you need to establish connectivity through private resources. For example, you can use the following strategies:

- Set up an AWS VPN or Direct Connect for on-premises access.
- Use a bastion host or VPN for administrative tasks from outside the VPC.

The decision between public and private access must align with your application architecture and security requirements. Public access can simplify connectivity during early development or testing, but private access is better for production workloads or sensitive data.

If you use the **Easy create** option when you create a DB instance, it uses private access by default. To change the access type, modify the DB instance. In the **Connectivity** section, expand **Additional configuration** and change the **Public access** setting.



For more information, see Settings for DB instances in the *Amazon RDS User Guide*.

Next steps

Now that you decided on public or private access for your DB instance, the next step is to define who can connect to your DB instance.

Next step: the section called “Determining who can connect”

Determining who can connect to your Amazon RDS DB instance

Security groups act as virtual firewalls for your Amazon RDS DB instance, controlling inbound and outbound traffic. It's important to properly configure security groups to ensure that only trusted connections can access your database.

Topics

Purpose of security groups and default behavior

Security groups in RDS determine which IP addresses and ports can connect to your DB instance. By default, each DB instance has the following behavior:

- Blocks all inbound traffic.
- Allows outbound traffic to all destinations.

To connect to your DB instance, you must explicitly define inbound rules that specify allowed IP addresses, protocols, and port numbers. This setup prevents unauthorized access and grants flexibility for trusted connections.

Configuring inbound and outbound rules

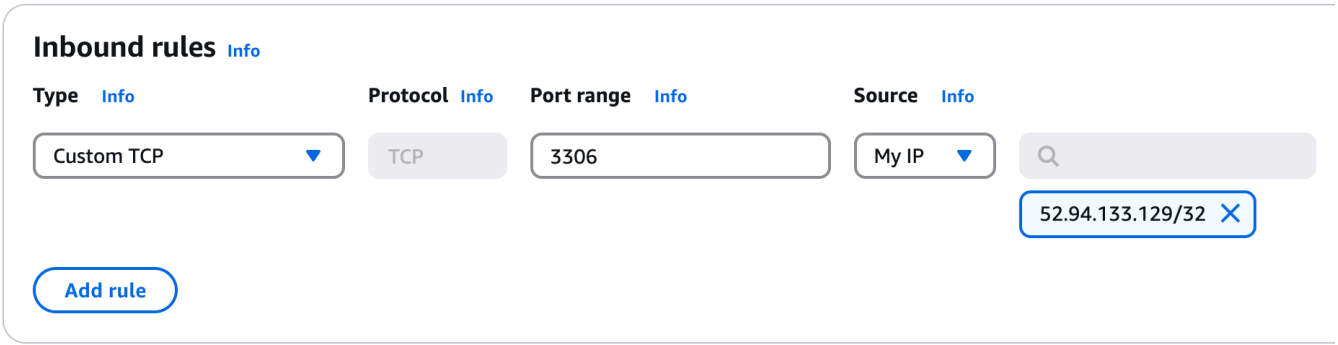
Follow these steps to configure inbound and outbound rules for your security group.

To configure inbound and outbound rules

1. Navigate to the **Amazon EC2 console** and choose **Security Groups** under the **Network & Security** menu.
2. Select or create a security group for your DB instance.
3. Configure inbound rules:
    - Add a rule to allow access on the database port (for example, port 3306 for MySQL) from a specific IP address or IP range.

- For the **Source**, choose **My IP** for personal access, or specify a CIDR block for broader access.

  For example, you might allow inbound traffic from an application hosted on an EC2 instance within the same VPC by specifying the EC2 instance security group as the source.



4. (Optional) Adjust outbound rules. Typically, you don't need to make any changes because the DB instance allows all outbound traffic by default.
5. Save your changes and associate the security group with your DB instance in the RDS console. Modify the DB instance and choose the new security group in the **Connectivity** section.

For more information, see Controlling access with security groups in the *Amazon RDS User Guide*.

Next steps

After you explicitly define at least one inbound rule that allows an IP address to access your DB instance, the next step is to decide on a database authentication option.

> **Note**
>
> The **Easy create** option for database creation uses password authentication by default, which is the simplest option. If you want to explore advanced authentication options, proceed to the next step. Otherwise, you can skip directly to connecting to your DB instance.

Next step: the section called "Database authentication options" or *Connecting to your DB instance*

Database authentication options for Amazon RDS

Authentication determines how users and applications connect to your Amazon RDS database. This section provides an overview of the available authentication options in Amazon RDS and guidance for selecting the right method for your use case.

This topic provides an overview of the available authentication options. For comprehensive documentation, see Database authentication with Amazon RDS in the *Amazon RDS User Guide*.

> **Note**
>
> By default, the **Easy create** option configures your database for password authentication. This means that you must specify a master username and password during creation, which are required when you connect to your DB instance. This is the simplest authentication option and requires no further configuration. If you need advanced authentication mechanisms, consider using the **Standard create** option.

Topics

Password authentication

Password authentication is the default mechanism for databases. This method requires users to connect by providing a database username and password.

- **Use case**: Simple and widely supported, suitable for basic database configurations and scenarios where external authentication systems are unnecessary.
- **Considerations**: Ensure strong password policies and secure storage of credentials.

IAM database authentication

IAM database authentication allows users and applications to connect to your database using AWS Identity and Access Management (IAM) users and roles in addition to the database password.

- **Use case**: Enhances security by leveraging IAM for access control and avoids storing credentials directly in your application.
- **Considerations**: Ensure strong password policies and secure storage of credentials.

Kerberos authentication

Kerberos authentication integrates with an existing Active Directory to authenticate users connecting to your DB instance.

- **Use case**: Enhances security by leveraging IAM for access control and avoids storing credentials directly in your application.
- **Considerations**: Requires IAM policies and roles to be correctly configured. Best suited for environments that already use IAM for identity management.

Next steps

Now that your DB instance is secure, the next step is to connect to it.

Next step: *Connecting to your DB instance*

Connecting to your Amazon RDS DB instance

Once your Amazon RDS DB instance is up and running, the next step is to establish a connection. This chapter starts by teaching you how to retrieve the necessary connection details directly from the AWS Management Console or using the AWS Command Line Interface (AWS CLI). From there, it walks through the process of connecting to your instance using popular database clients such as MySQL Workbench and pgAdmin.

The chapter also also addresses common connection issues and how to troubleshoot them. Finally, it describes how to test connectivity using the AWS CLI and verify your network and authentication configurations. By the end of this chapter, you'll be ready to connect to your Amazon RDS DB instance and start to manage your data.

Topics

Using the Amazon RDS console to retrieve connection information

Before you can connect to your Amazon RDS DB instance, you need to gather the connection details, including the endpoint, port, and other required settings. The AWS Management Console provides an easy way to retrieve this information. The following sections walk you through how to find the endpoint and port, along with additional connection details, so you can connect to your DB instance.

Topics

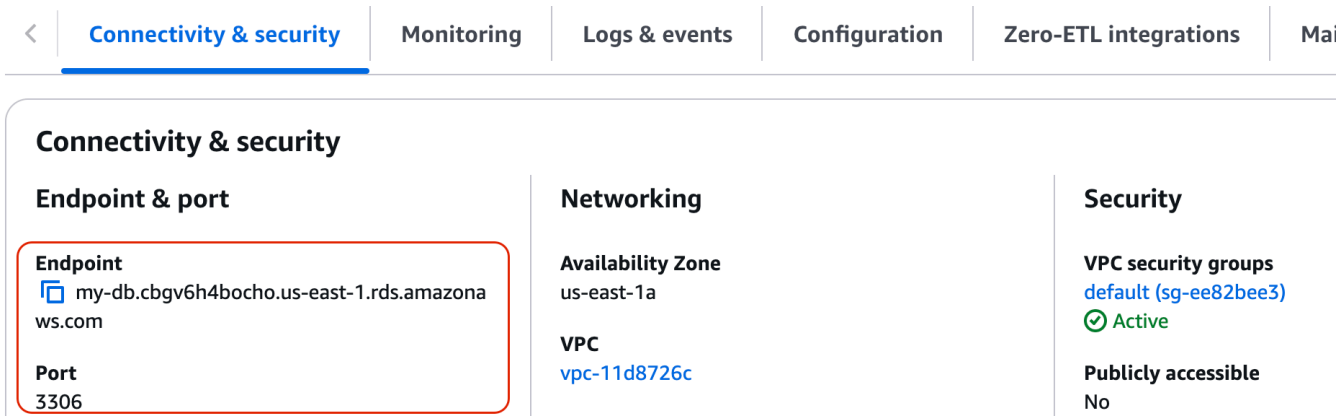
Locating the endpoint and port

To connect to your DB instance, you first need the instance endpoint and port number. Follow these steps to find them in the AWS Management Console.

To locate the endpoint and port

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the left navigation pane, choose **Databases**.
3. Select the DB instance that you want to connect to from the list of available instances.
4. In the **Connectivity & security** section, find the **Endpoint** and **Port** settings.
   - The **Endpoint** is the DNS address for your DB instance. You use this as part of the connection string when you connect with a database client.
   - The **Port** is the communication port used by the database engine (for example, 3306 for MySQL or 5432 for PostgreSQL).

     The following image shows these fields in the console:



These are the primary details that you need to initiate a connection to your DB instance.

Locating other connection details

In addition to the endpoint and port, you might need other connection details depending on your specific use case and database engine. You can find the following additional information in the **Connectivity & security** section and in the **Configuration** section.

- **VPC and subnet group** – The Virtual Private Cloud (VPC) and subnet group details help you understand the network environment your DB instance resides in. If you need to configure security groups or modify network settings, this information is essential.
- **Security groups** – Security groups control access to your DB instance. You can view the security groups associated with your DB instance here, which help you make sure that the appropriate inbound and outbound rules are in place for a successful connection.
- **DB parameter group** – If you need to adjust database settings, such as timeouts or query limits, the DB parameter group associated with your instance provides the necessary configuration options.

Next steps

Once you have the required connection details, including the instance endpoint and port, you can use them to connect to the DB instance.

Next step: the section called “Connecting using a database client”

Using the AWS CLI to retrieve and validate connection information for Amazon RDS

While the AWS Command Line Interface (AWS CLI) doesn't directly connect to a database for querying or interactive use, it provides tools to manage and test the connectivity of your Amazon RDS DB instance. You can retrieve connection details, validate network configurations, and generate authentication tokens if you're using IAM authentication. This section explains how to use the AWS CLI to prepare to connect to your database using a client tool.

Testing connectivity using the AWS CLI

Test connectivity with the AWS CLI to make sure you can reach your DB instance from your local environment. The following steps guide you through retrieving and verifying connection details.

To test connectivity using the CLI

1. Retrieve the endpoint and port. Use the following describe-db-instances command to retrieve connection information for your DB instance:

```
aws rds describe-db-instances \
  --db-instance-identifier your-db-instance-id \
```

```
  --query "DBInstances[0].[Endpoint.Address, Endpoint.Port]"
```

Replace `your-db-instance-id` with the identifier of your DB instance.

The output should include the endpoint (hostname) and port, which you need in order to configure your database client.

2. Verify the output. Make sure that the endpoint and port match the values in the AWS Management Console. If there are any discrepancies, check the DB instance configuration.

Resolving network and authentication issues using the AWS CLI

Misconfigured network settings or incorrect authentication credentials are common causes of connectivity issues. The AWS CLI provides commands to check and resolve these problems.

1. Check network settings. Network issues often stem from incorrect security group or VPC configurations.

   - **Verify security group rules**. Make sure that the security group associated with your DB instance allows inbound traffic on the database port (for example, 3306 for MySQL or 5432 for PostgreSQL) from your IP address.

     ```
     aws ec2 describe-security-groups \
       --group-ids security-group-id
     ```

     Replace `security-group-id` with the ID of your security group.

   - **Check subnet configurations**. Confirm that your DB instance resides in a subnet that allows connectivity.

     ```
     aws ec2 describe-subnets \
       --subnet-ids subnet-id
     ```

     Replace `subnet-id` with the ID of the subnet for your DB instance.

2. Validate configuration details. Authentication errors can occur due to incorrect credentials or improper IAM configuration.

   - **Reset the master password**. If the master password is incorrect or you forgot it, reset it using the following moidfy-db-instance command:

```
aws rds modify-db-instance \
  --db-instance-identifier db-instance-id \
  --master-user-password new-password
```

Replace `db-instance-id` with your instance ID and `new-password` with the new password.

- **Verify the master username**. Confirm the master username for your DB instance with the following describe-db-instances command:

```
aws rds describe-db-instances \
  --db-instance-identifier db-instance-id \
  --query "DBInstances[0].MasterUsername"
```

Make sure that the username matches what you're using in your database client.

- **Check IAM authentication**. If your DB instance uses IAM authentication, generate a temporary token for login with the following generate-db-auth-token command:

```
aws rds generate-db-auth-token \
  --hostname endpoint \
  --port port \
  --username iam-user
```

Replace `endpoint`, `port`, and `iam-user` with your DB instance endpoint, port, and IAM username. For more information, see IAM database authentication for MariaDB, MySQL, and PostgreSQL in the *Amazon RDS User Guide*.

By combining these CLI commands, you can verify that your DB instance is accessible and correctly configured. If connection issues persist, see Troubleshooting for Amazon RDS in the *Amazon RDS User Guide*.

Next steps

Once you have the required connection details, including the instance endpoint and port, you can use them to connect to the DB instance.

Next step: the section called "Connecting using a database client"

Connecting to an Amazon RDS DB instance using a database client

Database clients provide a user-friendly way to connect to and manage your Amazon RDS DB instance. This section outlines the process of connecting to your DB instance using two popular database clients: MySQL Workbench and pgAdmin.

Althought the exact steps vary slightly depending on the tool and database engine, the general process involves configuring the connection with your endpoint, port, and credentials.

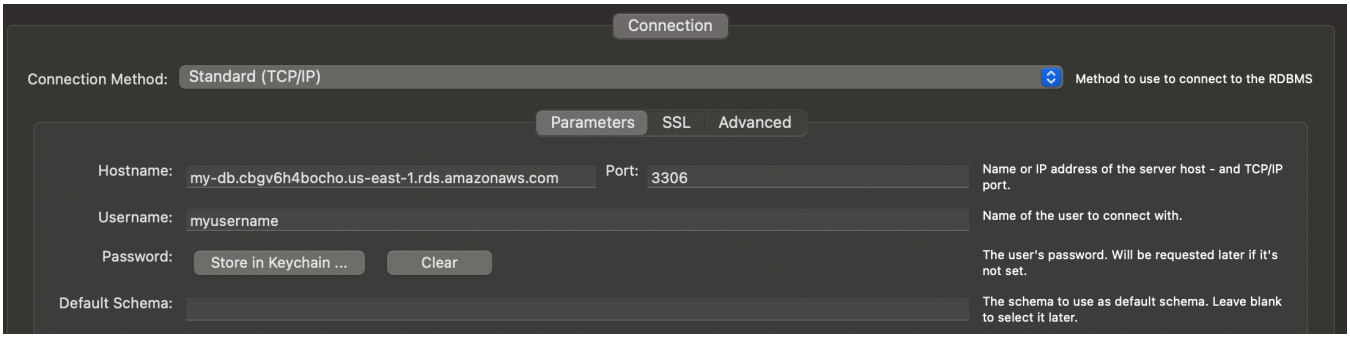
Topics

Connecting to a MySQL DB instance

MySQL Workbench is a popular database client that allows you to connect to and manage your MySQL DB instance. Follow these steps to set up a connection and start working with your MySQL database.

To connect to a MySQL DB instance

1. Open MySQL Workbench on your local machine.
2. Choose **Database**, **Manage Connections** from the menu.
3. Create a new connection and configure the following settings:
    - **Hostname**: Enter the endpoint retrieved from the AWS Management Console.
    - **Port**: Use the port number displayed in the **Connectivity & security** section (typically 3306).
    - **Username**: Enter the master username you set when you created the DB instance.



4. Choose **Test Connection** to verify the connection settings.
5. When the connection is successful, save the configuration and open the connection to access your database.

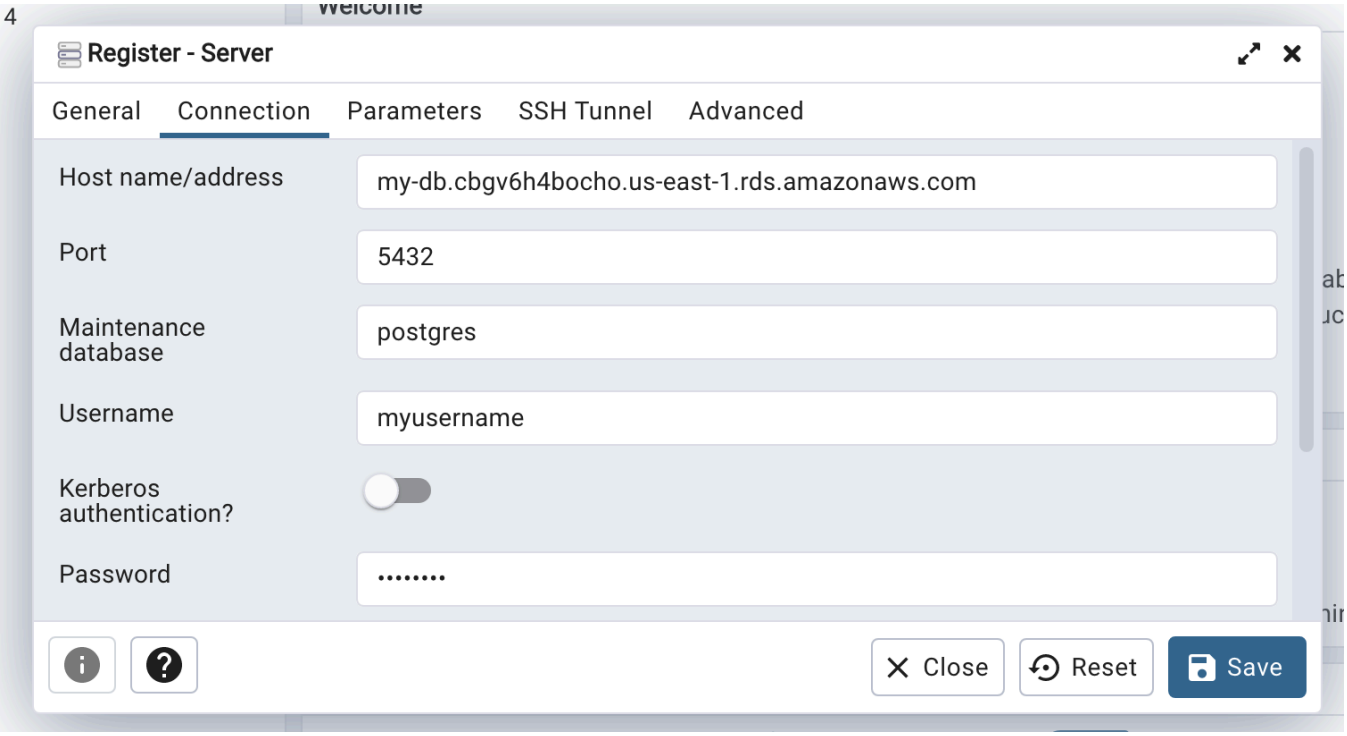
For comprehensive documentation, see Connecting to a DB instance running the MySQL database engine in the *Amazon RDS User Guide*.

Connecting to a PostgreSQL DB instance

pgAdmin is a comprehensive management tool for PostgreSQL databases that simplifies connecting to and administering your RDS for PostgreSQL DB instance. Use the following steps to configure your connection and interact with your database.

To connect to a PostgreSQL DB instance

1. Launch pgAdmin on your system.
2. Choose **Add New Server**.
3. In the **General** tab, enter a name for the connection. For example, "My RDS instance".
4. In the **Connection** tab, configure the following settings:
   - **Host**: Enter the endpoint from the AWS Management Console.
   - **Port**: Use the port number provided (typically 5432).
   - **Username**: Enter the master username for your DB instance.
   - **Password**: Provide the password you set during instance creation.



5. Save the configuration and connect in order to view and manage your database.

For comprehensive documentation, see Connecting to a DB instance running the PostgreSQL database engine in the *Amazon RDS User Guide*.

Connecting to other database engines

In addition to PostgreSQL and MySQL, Amazon RDS supports several other database engines. To connect to these databases, see the following documentation in the *Amazon RDS User Guide*.

- **MariaDB**: Connecting to your MariaDB DB instance
- **Microsoft SQL Server**: Connecting to your Microsoft SQL Server DB instance
- **Oracle**: Connecting to your Oracle DB instance
- **IBM Db2**: Connecting to your Db2 DB instance

Each database engine has specific requirements and configuration options. These topics provide instructions to help you establish a secure connection to your DB instance.

For a comprehensive overview of all supported database engines and their features, see the Amazon RDS features.

Next steps

At this stage, you have successfully created and connected to your RDS DB instance. From here, you can explore management strategies such as backing up, monitoring, optimizing, and scaling your DB instance.

Additionally, consider reviewing resources that provide practical guidance on advanced configurations, performance tuning, security enhancements, and cost management strategies.

Next steps:

- *Managing your DB instance*
- *Optimizing and scaling*
- *Additional resources*

Troubleshooting connection issues to your Amazon RDS DB instance

When you attempt to connect to an Amazon RDS DB instance, you might encounter common issues that prevent successful connections. This topic addresses several frequent connection problems, along with steps to identify and resolve them.

Topics

Incorrect security group configuration

If the security group associated with your DB instance doesn't allow traffic from your client, connections will fail.

Solution:

- Verify the security group rules in the Amazon EC2 console.
- Ensure inbound rules allow traffic on the database port (3306 for MySQL, 5432 for PostgreSQL, and so on).
- Add your client IP address or a CIDR block to the inbound rules.

For more information, see Controlling access with security groups.

Incorrect database endpoint and port

Using the wrong endpoint or port results in failed connection attempts.

Solution:

- Retrieve the correct endpoint from the RDS console.
- Make sure you're using the database's assigned port.
- Check for typos in the connection string.

For more information, see Finding the connection information for an RDS for MySQL DB instance.

Network ACLs blocking traffic

If Network Access Control Lists (NACLs) block traffic to or from the subnet, connection attempts fail.

Solution:

- Check the NACLs associated with your subnet in the Amazon VPC console.
- Make sure that inbound and outbound rules allow traffic on your database port.

For more information, see Control subnet traffic with network access control lists.

Authentication errors

Using incorrect credentials or configuration errors in database authentication can result in failed logins.

Solution:

- Confirm the username and password in your connection string.
- Check IAM policies if you're using IAM authentication.

For more information, see IAM database authentication for MariaDB, MySQL, and PostgreSQL .

VPC peering or network misconfigurations

Misconfigured peering connections or route tables might block communication between the client and the database.

Solution:

- Verify that the VPC peering connection is active.
- Check route tables to ensure traffic can flow between VPCs.
- Make sure there are no overlapping IP ranges between VPCs.

For more information, see Connect VPCs using VPC peering.

Next steps

If these steps don’t resolve your connection issues, consider enabling enhanced logging or contacting Support for further assistance. Additionally, explore the troubleshooting guides specific to your database engine:

- Troubleshooting connections to your MySQL DB instance
- Troubleshooting connections to your PostgreSQL DB instance

Managing your Amazon RDS DB instance

Effective database management ensures data availability, reliability, and performance over time. This chapter covers best practices for managing your Amazon RDS DB instance, with a focus on backup and restore capabilities. These strategies help safeguard your data and maintain business continuity.

Topics

Backing up and restoring your Amazon RDS DB instance

Data backups are important for recovering from accidental deletions, corruption, or unexpected failures. Amazon RDS provides both automated backups and manual snapshots, so you have flexibility in how you protect and restore your data. This section provides an overview of these options and practical guidance for using them.

Topics

Automated backups

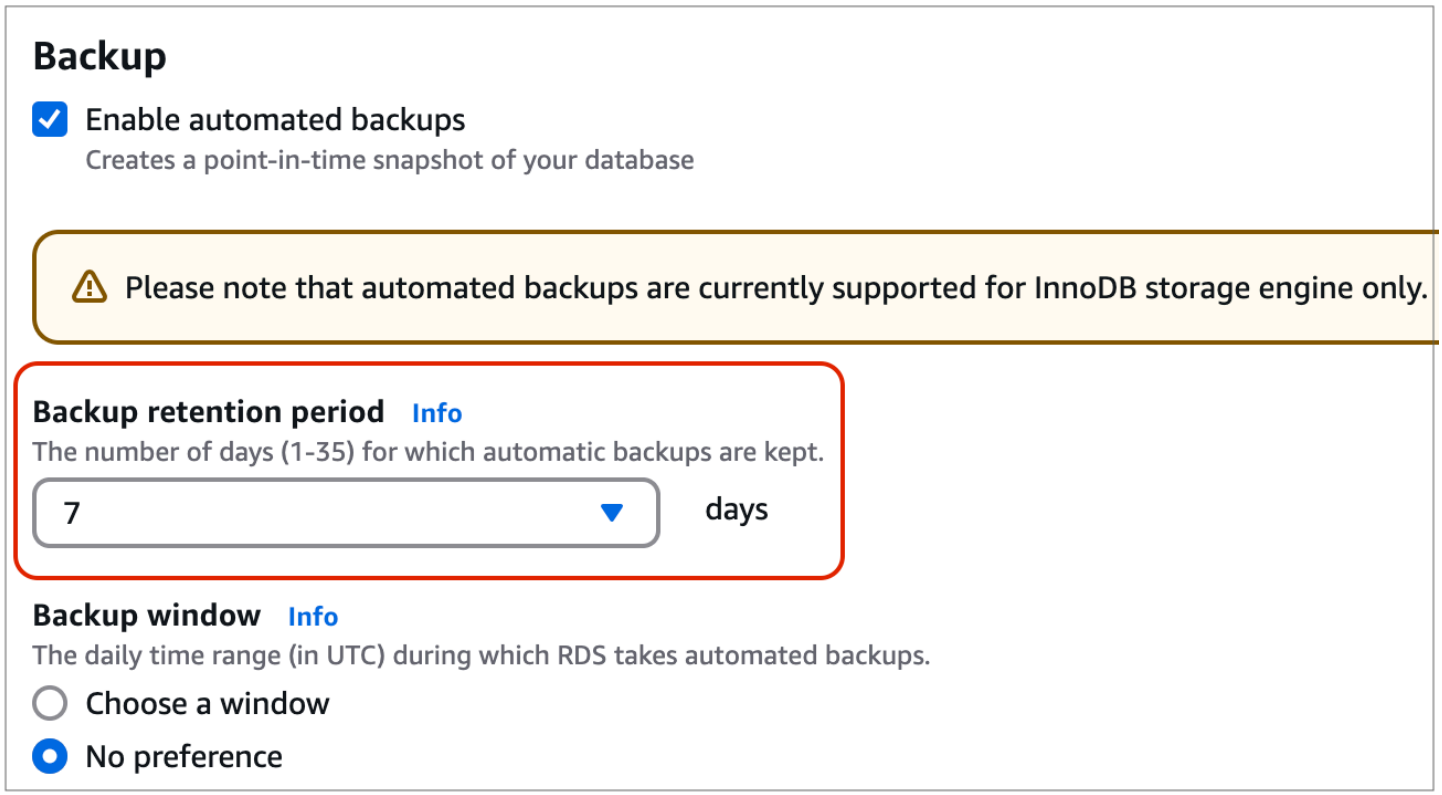
Automated backups in Amazon RDS provide a reliable way to protect your data by creating backups of your database at regular intervals and retaining them for a specified period. Amazon RDS enables this feature by default when you create your DB instance. It ensure that you can recover to almost any point within the retention period without manual intervention.

These sections provide a brief overview of the available backup strategies. For comprehensive documentation on automated backups, see Managing automated backups in the *Amazon RDS User Guide*.

Retention periods

The retention period determines how long Amazon RDS retains automated backups before it automatically deletes them. You can configure this period to meet your compliance or operational requirements, with options ranging from 1 to 35 days.

To configure how long Amazon RDS retains automated backups, set the **Backup retention period** setting for the DB instance:



Consider the following when you configure a retention period:

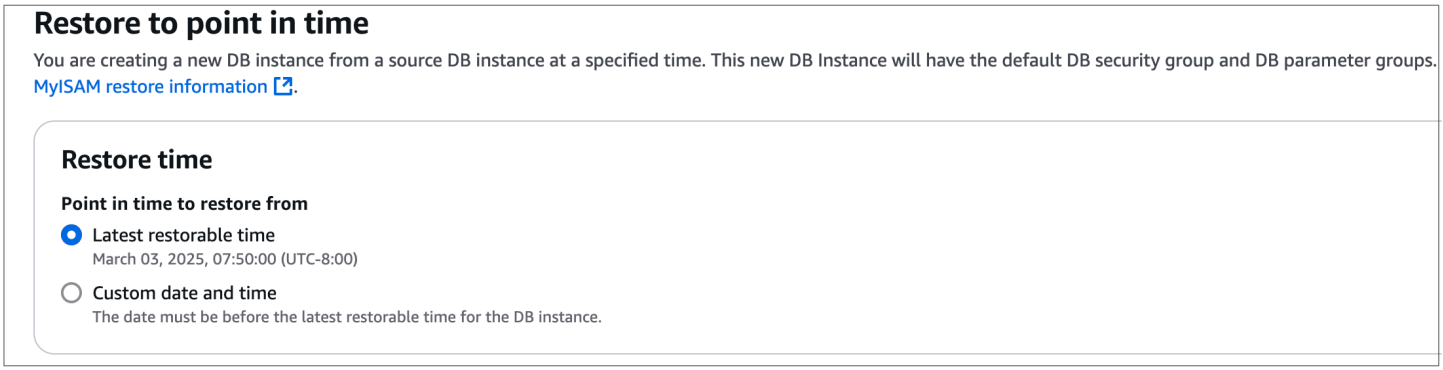
- A longer retention period provides more recovery options but might increase storage costs.
- RDS stores backups in Amazon S3, and AWS calculates the storage cost separately from your DB instance usage.

For more information, see Retaining automated backups in the *Amazon RDS User Guide*.

Point-in-time recovery

With point-in-time recovery, you can restore your database to any second within your backup retention period. This is especially useful in scenarios such as accidental data deletion or corruption.

To perform a point-in-time recovery, choose **Automated backups** within the Amazon RDS console and select the DB instance that you want to restore. Then, choose **Actions**, **Restore to point in time**. Specify the exact time to which you want to restore your database. Amazon RDS creates a new instance from the backups and leaves the original instance intact.



For more information, see Restoring a DB instance to a specified time for Amazon RDS in the *Amazon RDS User Guide*.

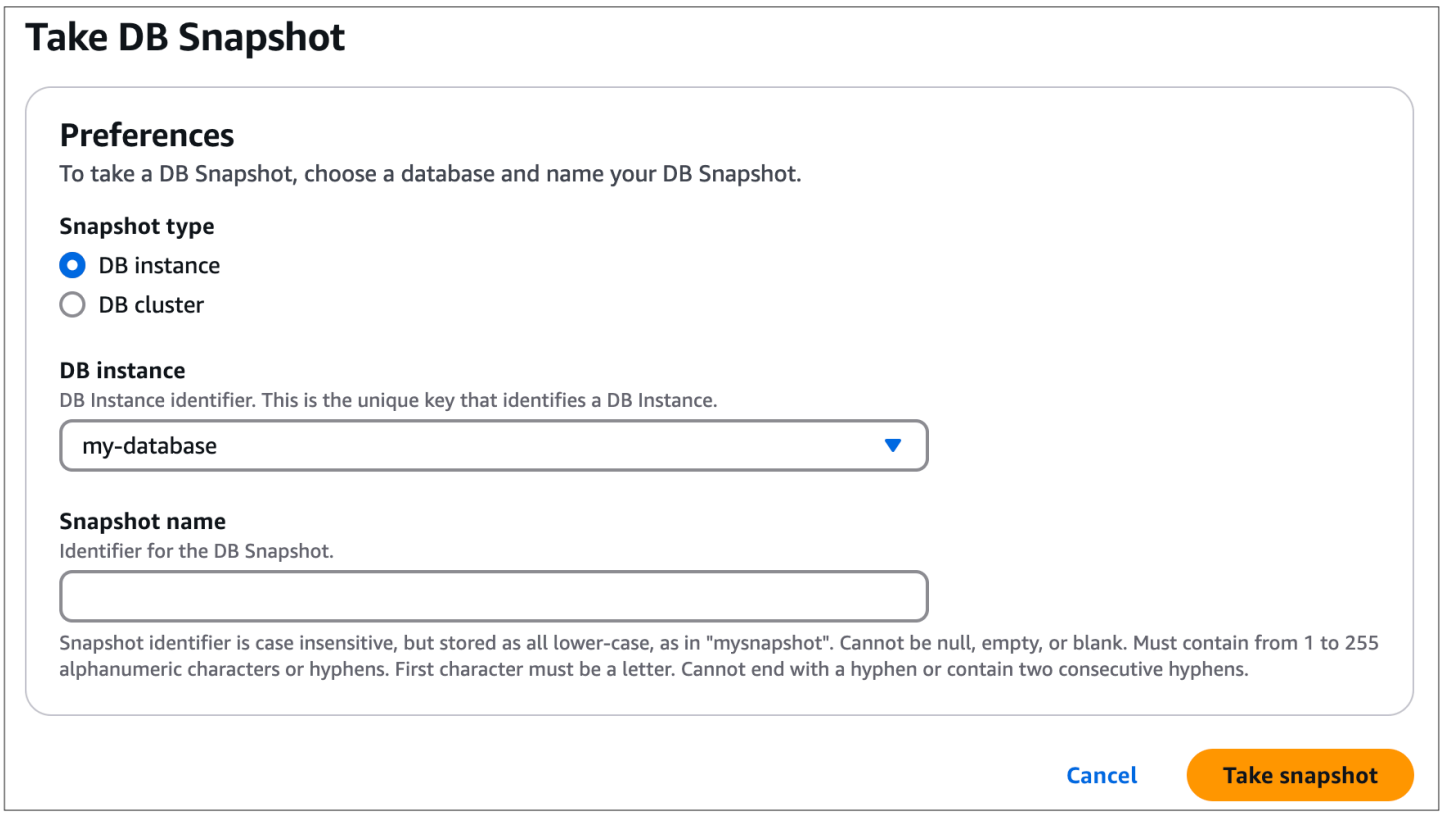
Manual snapshots

Manual snapshots provide more control over your backup strategy, and let you capture a complete copy of your database at a specific point in time. Unlike automated backups, manual snapshots are user-initiated and persist until you explicitly delete them. This provides a long-term data retention option.

Manual snapshots have the following main benefits:

- Ideal for preserving data before significant schema changes or updates.
- Useful for duplicating a database in a different environment, such as development or testing.
- Provide an additional layer of redundancy when paired with automated backups.

To take a manual snapshot, select the DB instance that you want to back up and choose **Actions**, **Take snapshot**. RDS stores manual snapshots in Amazon S3. You can share them across AWS accounts or copy them to different AWS Regions for disaster recovery purposes.



Restoring from a manual snapshot involves creating a new DB instance from the stored backup. Choose **Snapshots** within the Amazon RDS console and select the snapshot that you want to restore. Choose **Actions**, **Restore snapshot**. Specify the instance details for the new database.

For more information, see Creating a DB snapshot for a Single-AZ DB instance for Amazon RDS in the *Amazon RDS User Guide*.

Monitoring your Amazon RDS DB instance

Effective monitoring is essential to maintain the health and performance of your Amazon RDS DB instance. AWS provides multiple tools and features to help you track key metrics, identify potential issues, and optimize your database performance. This section introduces the primary monitoring tools available in Amazon RDS and explains how they address different aspects of database health and performance.

Topics

- Monitoring with Amazon CloudWatch
- Monitoring with Amazon CloudWatch Logs
- Using Enhanced Monitoring

- Monitoring with Performance Insights
- Monitoring with Database Activity Streams

Monitoring with Amazon CloudWatch

Amazon CloudWatch collects and tracks metrics related to your DB instance, such as CPU utilization, storage usage, and read/write operations. You can create alarms and dashboards for real-time visibility into the status of your database.

Key features:

- Tracks over 50 default metrics for Amazon RDS DB instances.
- Provides historical data retention for trend analysis.
- Lets you set alarms to notify you when thresholds are breached.

Use cases:

- Monitor storage utilization to ensure adequate capacity.
- Detect unusual spikes in CPU or memory usage.
- Track connection counts to optimize application scaling.

For more information, see Monitoring Amazon RDS metrics with Amazon CloudWatch in the *Amazon RDS User Guide*.

Monitoring with Amazon CloudWatch Logs

Amazon RDS provides database logs that help you diagnose issues, analyze performance, and track security-related events. You can access logs through the RDS console, the AWS CLI or the RDS API and enable monitoring with Amazon CloudWatch Logs.

Key features:

- Supports error logs, slow query logs, and general logs for troubleshooting.
- Integrates with CloudWatch Logs for real-time analysis and alerting.
- Allows download and viewing of logs directly from the RDS console.

Use cases:

- Identify and troubleshoot database errors by analyzing error logs.
- Monitor slow query logs to optimize database performance.
- Track authentication attempts and security-related events in audit logs.

For more information, see Monitoring Amazon RDS log files in the *Amazon RDS User Guide*.

Using Enhanced Monitoring

Enhanced Monitoring provides detailed, real-time metrics about the operating system running on your Amazon RDS DB instance. It collects these metrics at a granularity of up to one second and give you deeper insight into the underlying hardware and software performance.

Key features:

- Monitors over 50 OS-level metrics, including disk I/O, swap usage, and processes.
- Provides a more granular view of system resource usage compared to CloudWatch.
- RDS can export data to CloudWatch Logs for further analysis.

Use cases:

- Diagnose performance bottlenecks related to disk or memory.
- Gain insights into background processes consuming resources.
- Monitor the impact of maintenance tasks or application updates.

For setup instructions, see Monitoring OS metrics with Enhanced Monitoring in the *Amazon RDS User Guide*.

Monitoring with Performance Insights

Performance Insights is a tool for analyzing database performance and identifying bottlenecks. It provides a visual interface that highlights resource consumption trends and offers insights to optimize your queries and workloads.

Key features:

- Monitors database load in real time and over historical periods.
- Identifies top SQL queries consuming resources.
- Visualizes wait events to pinpoint delays in database processing.

Use cases:

- Optimize queries that consume excessive resources.
- Identify and address high latency due to locking or I/O issues.
- Plan capacity upgrades by analyzing historical load trends.

Most database engines support Performance Insights, but its availability varies by engine and instance type. For more information, see Monitoring DB load with Performance Insights on Amazon RDS in the *Amazon RDS User Guide*.

Monitoring with Database Activity Streams

Database Activity Streams provide a real-time stream of database activity, which helps you monitor and audit database operations for security and compliance. You can integrate activity streams with AWS services like Amazon CloudWatch and AWS Key Management Service (AWS KMS) for encryption.

Key features:

- Captures database activity in near real-time for security monitoring.
- Encrypts all activity streams using AWS KMS.
- Integrates with Amazon CloudWatch and third-party security tools for analysis.

Use cases:

- Detect and investigate suspicious or unauthorized database activity.
- Maintain compliance by auditing changes and access patterns.
- Monitor SQL commands and administrative actions for security insights.

For more information, see Monitoring Amazon RDS with Database Activity Streams in the *Amazon RDS User Guide*.

Optimizing and scaling your Amazon RDS DB instance

This chapter includes details on advanced configurations and strategies for optimizing and scaling your Amazon RDS DB instance. These practices help ensure your database remains highly available, performs efficiently under varying workloads, and supports your business needs. It also guides you through key concepts for achieving high availability, scaling horizontally, and adapting to changing performance demands.

Topics

Scaling and high availability in Amazon RDS

Scaling and high availability are important aspects of managing your Amazon RDS DB instance to meet changing application requirements. Scaling helps your database handle increased workloads, while high availability ensures minimal disruption during unexpected failures or maintenance events.

This topic highlights features such as instance resizing, read replicas, and Multi-AZ deployments to improve performance and resilience.

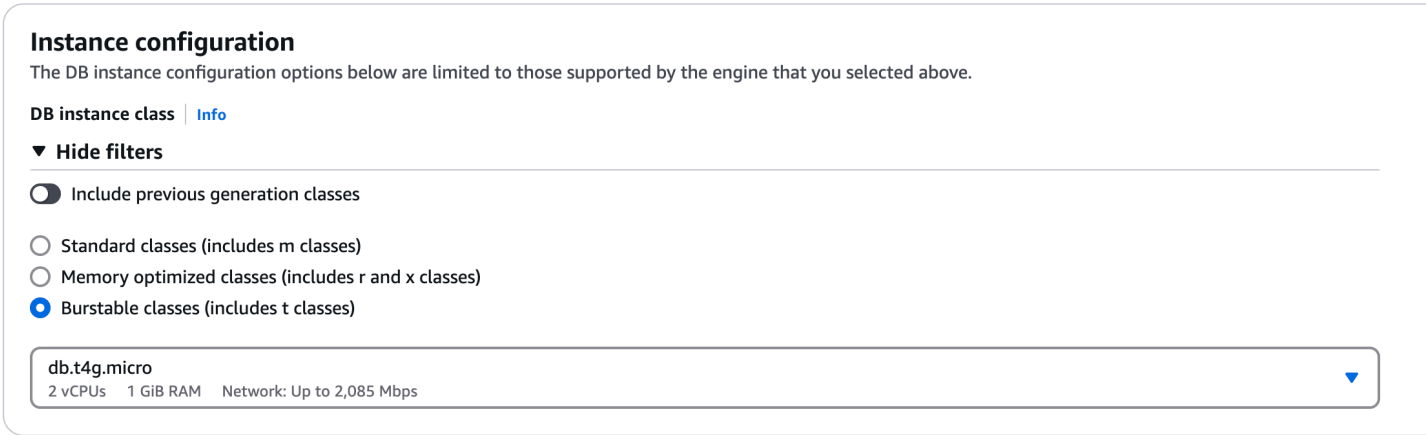
Topics

Changing instance sizes to adjust for demand

With Amazon RDS, you can scale vertically by modifying your DB instance size to meet your performance and capacity requirements. Resizing an instance helps accommodate changes in application demand, such as increased traffic or computational needs.

To change the instance size for your DB instance, navigate to the DB instance details page in the AWS Management Console and choose **Modify**. For the **Instance configuration**, choose a

new instance class that aligns with your requirements, such as compute optimized or memory optimized.



Apply changes during a maintenance window or immediately, depending on your application's tolerance for downtime.

Key considerations:

- Make sure the instance class that you choose supports your storage and workload requirements.
- Larger instance sizes often come with increased costs, so evaluate your budget before scaling.

For comprehensive documentation, see DB instance classes in the *Amazon RDS User Guide*.

Adding read replicas for load distribution

Amazon RDS provides horizontal scaling by using read replicas to offload read-intensive workloads and improve database performance. Read replicas are copies of your primary database that asynchronously replicate data. They're ideal for scaling read-heavy applications and improving response times. Read replicas distribute read traffic across multiple instances, and provide additional redundancy for disaster recovery scenarios.

To create a read replica for an existing DB instance, select it within the AWS Management Console. From the **Actions** menu, choose **Create read replica**, then configure the instance class and AWS Region for the replica. Launch the replica and update your application to direct read traffic accordingly.

Key considerations:

- Read replicas are read-only and can't handle write operations.

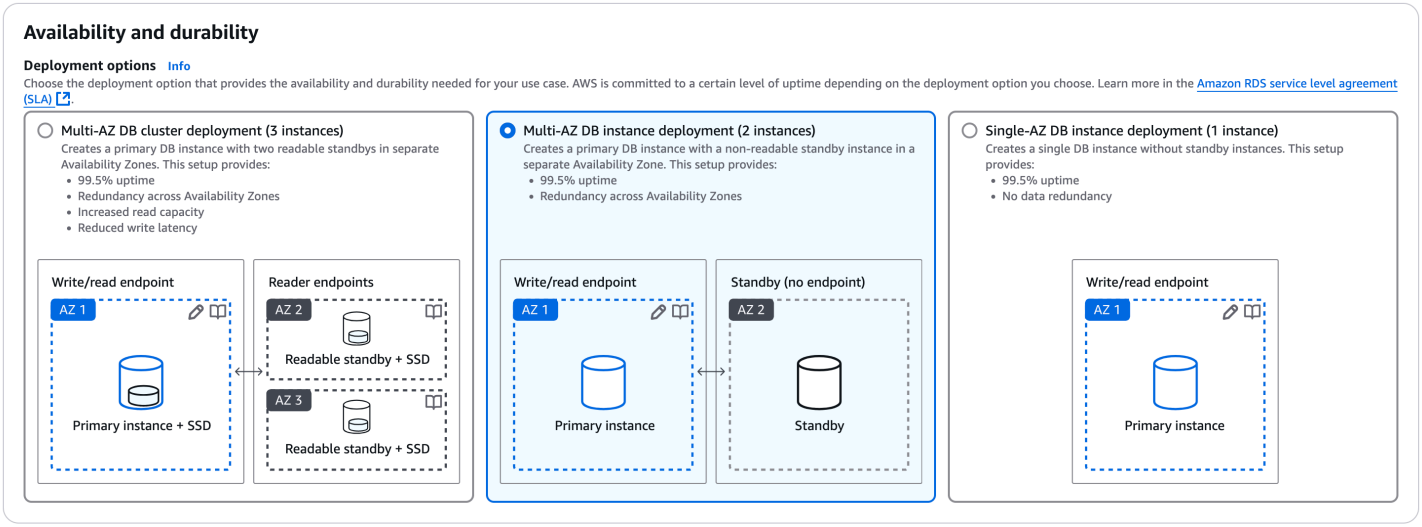
- Replicas might introduce minor latency due to asynchronous replication.

For comprehensive documentation, see Working with DB instance read replicas in the *Amazon RDS User Guide*.

Enabling Multi-AZ for high availability

Multi-AZ deployments provide high availability and durability by automatically maintaining a synchronous standby replica in a different Availability Zone. Multi-AZ deployments minimize downtime during planned maintenance or infrastructure failure by automatically failing over to the standby replica.

To create a Multi-AZ DB instance, select **Multi-AZ DB instance** under **Availability and durability** when you create the DB instance. Amazon RDS handles data replication and failover management transparently.



Key considerations:

- Multi-AZ deployments increase storage requirements, which might impact storage costs and capacity planning.
- Failover between Availability Zones might result in a brief interruption, so configure your applications to handle reconnections.

For comprehensive documentation, see Multi-AZ DB instance deployments for Amazon RDS in the *Amazon RDS User Guide*.

Advanced security options in Amazon RDS

It's important to secure your Amazon RDS DB instance to protecting your data and ensuring compliance with organizational and regulatory standards. This section covers advanced network configurations and encryption techniques to improve your database security.

Topics

Accessing a DB instance in a VPC

When you host an Amazon RDS DB instance within an Amazon Virtual Private Cloud (Amazon VPC), you must understand the various networking options available to securely connect to your database. By configuring the VPC environment appropriately, you can access your DB instance from application servers, on-premises systems, or other VPCs.

For example, consider an application running in an on-premises data center that needs to connect to a DB instance in a VPC. Using Direct Connect or a VPN connection, you can establish a private link to securely access the database while keeping the communication isolated from the public internet.

The following are the main ways to connect to a DB instance:

- **Within the same VPC** – Applications running on instances within the same VPC can communicate directly with the DB instance using its private IP address. Make sure that appropriate security group and network ACL rules allow the necessary traffic.
- **From a different VPC** – If your application is in another VPC, establish connectivity through VPC peering, AWS Transit Gateway, or private VPC endpoints. Each option has unique benefits and cost implications.
- **From on-premises systems** – Use Direct Connect or a VPN connection to create a secure link between your on-premises environment and the VPC hosting the DB instance.
- **From public internet** – If you enable public access, you can connect to the DB instance using its public endpoint. Carefully manage IP whitelisting and encryption in order to maintain security.

Key considerations:

- Use security groups to define which traffic can reach the DB instance.
- Restrict access by IP address to minimize the exposure of your DB instance.
- When you use public access, enable Secure Sockets Layer and Transport Layer Security (SSL/TLS) for encrypted communication.

For detailed networking scenarios and configurations, see Scenarios for accessing a DB instance in a VPC in the *Amazon RDS User Guide*.

Enabling SSL/TLS encryption for data in transit

To secure communication between your application and an Amazon RDS DB instance, Secure Sockets Layer and Transport Layer Security (SSL/TLS) encryption protects data transmitted over the network from interception and tampering. This method is particularly relevant for applications where data must remain confidential during transit, such as financial records, personal information, or other sensitive content.

For example, a microservices architecture might include a reporting service that queries a database hosted in Amazon RDS. Without encryption, malicious actors or network intermediaries can intercept and potentially modify the data sent across the network. Enable SSL/TLS to encrypt these interactions and maintain data integrity and confidentiality.

To enable encryption in transit

1. Obtain the required certificates by downloading the Amazon RDS SSL/TLS certificates. See Certificate bundles by AWS Region in the *Amazon RDS User Guide*.
2. Adjust your database connection settings to include the certificate and enable SSL/TLS. This might involve specifying parameters in the connection string, such as `sslmode=require`, depending on the database client.
3. Confirm that the application connects to the DB instance securely by reviewing logs or using diagnostic tools to inspect the connection properties.

Key considerations:

- Use the most up-to-date version of TLS that your applications and RDS DB instance support, so you can use the the latest security features.

- Plan for certificate rotation by monitoring expiration dates and updating applications as needed.
- Test the configuration in a staging environment before you apply it in production in order to avoid service disruptions.

For implementation details, see to the Using SSL/TLS to encrypt a connection to a DB instance or cluster in the *Amazon RDS User Guide*.

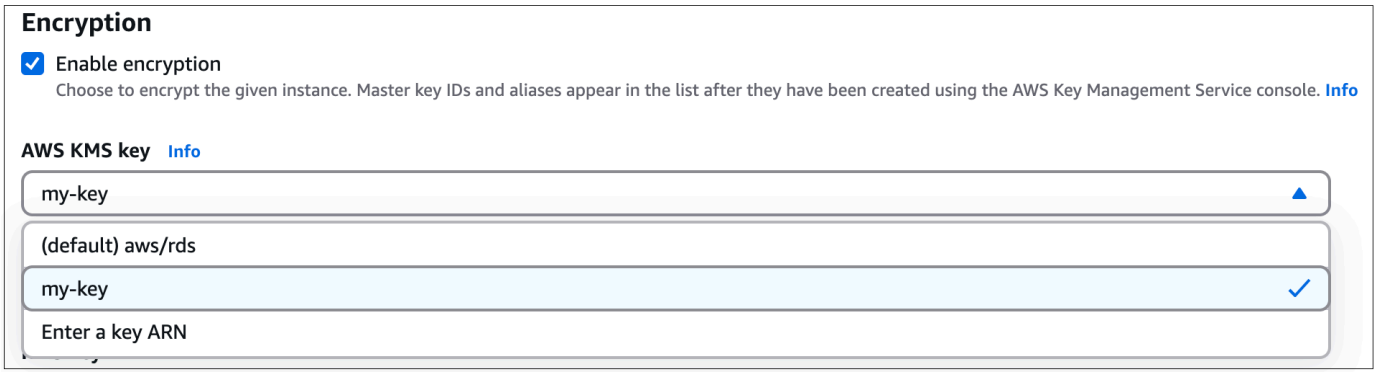
Encryption at rest with customer managed keys

With Amazon RDS, you can encrypt your database and backups to ensure that sensitive data remains secure when you store it. By using AWS Key Management Service (AWS KMS) with customer managed keys, you have full control over the encryption process. This includes the ability to define key rotation policies and restrict access to specific users or services.

When you enable encryption at rest, Amazon RDS encrypts all data in your database, including automated backups, read replicas, and snapshots, using the specified KMS key. This encryption occurs automatically, with minimal performance overhead for most workloads.

To enable encryption at rest

1. Create a KMS key. In the AWS Management Console, navigate to AWS KMS and create a customer managed key. Specify usage policies and rotation settings. For instructions, see Create a KMS key in the *AWS Key Management Service Developer Guide*.
2. When you launch a new Amazon RDS instance, select the option to enable encryption and specify your KMS customer managed key.



To encrypt existing databases, create a new encrypted DB instance or snapshot and migrate your data.

Key considerations:

- Use IAM policies to tightly restrict who can use or manage the encryption keys.
- Enable automatic key rotation for customer managed keys to simplify security management.
- Make sure that encryption aligns with your long-term requirements, because you can't disable encryption at rest after you enable it.

For more detailed guidance on setting up encryption at rest with AWS KMS, see Encrypting Amazon RDS resources in the *Amazon RDS User Guide*.

Additional resources for Amazon RDS

Amazon RDS offers multiple tools, configurations, and best practices to help you optimize your database management. To deepen your understanding of Amazon RDS, you can explore additional resources that provide practical guidance on advanced configurations, performance tuning, security enhancements, and cost management strategies. These resources can help you scale, secure, and manage your databases for diverse workloads and organizational needs.

This topic provides links to additional resources, including technical guides, tutorials, videos, and webinars, to further enhance your skills and knowledge. The materials can help you learn advanced features, adhere to industry best practices, and gain hands-on experience.

Topics

Advanced configurations

See the following resources to deepen your understanding of Amazon RDS advanced configurations:

- **Multi-AZ deployments**: Learn how to configure and maintain Multi-AZ deployments for enhanced availability. Read the guide.
- **Read replicas for load distribution**: Set up read replicas to improve performance and scale read workloads. Learn more.
- **Parameter groups**: Customize your database settings using parameter groups. Learn about parameter groups.
- **VPC scenarios**: Understand how to configure access and security for your DB instance in a VPC. Explore scenarios.
- **Storage**: Review storage options and performance considerations. Read the storage guide.

Best practices

Apply these best practices to enhance your database management:

- **Amazon RDS best practices**: A comprehensive guide to running Amazon RDS efficiently and securely. Explore best practices.
- **AWS Well-Architected framework**: Principles to design secure, high-performing, and resilient workloads. Read the framework.
- **Backup and restore best practices**: Ensure data durability with effective backup strategies. Learn about backups.
- **Security best practices**: Protect your database with recommended security measures. Review security best practices.

Tutorials and videos

Learn by doing with these tutorials and videos:

- **Tutorials**: Follow step-by-step tutorials to get started with Amazon RDS. Explore tutorials.
- **Blogs, videos, and whitepapers**: Find technical blogs, demo videos, and in-depth whitepapers. Access resources.
- **Hands-on labs**: Practice RDS configurations and deployments in a lab environment. Access labs.
- **AWS Online Tech Talks**: Join webinars hosted by AWS experts to learn about the latest features and best practices. Explore tech talks.

Document history for the Amazon Relational Database Service Getting Started Guide

The following table describes the documentation releases for the Amazon Relational Database Service Getting Started Guide.

Change	Description	Date
Initial release	Initial release of the Amazon Relational Database Service Getting Started Guide	March 5, 2025