Data Encryption in Amazon Aurora

**Contents**

[Scope 2](#_Toc115190395)

[Problem Statement 2](#_Toc115190396)

[Introduction 2](#_Toc115190397)

[Data Encryption 3](#_Toc115190396)

[Encrypting Amazon RDS Resources 3](#_Toc115190398)

[Overview of encrypting Amazon Aurora resources 3](#_Toc115190399)

[Encrypting a DB Instance 3](#_Toc115190398)

[Determining whether encryption is turned on for a DB cluster 4](#_Toc115190398)

[Availiability of Amazon Aurora Encryption 5](#_Toc115190398)

[Encrypting an Amazon Aurora DB cluster 5](#_Toc115190400)

[Type of Encryption used in Amazon Aurora 6](#_Toc115190401)

[AES-256 Encryption Algorithm 6](#_Toc115190401)

[Envelope Encryption 6](#_Toc115190401)

[KMS Key Management 9](#_Toc115190396)

[Authorising use of a customer managed key 10](#_Toc115190401)

[Using SSL/TLS to encrypt a connection to DB cluster 11](#_Toc115190413)

[Rotating your SSL/TLS certificate 12](#_Toc115190413)

[Updating CA certificate by modifying DB instance 14](#_Toc115190401)

[Updating CA certificate by applying DB instance maintenance 15](#_Toc115190401)

[Updating CA certificate by applying maintenance to multiple DB instances 18](#_Toc115190401)

[Automation of back ups 29](#_Toc115190413)

[Database Authentication 31](#_Toc115190413)

[Password Authentication 31](#_Toc115190401)

[IAM Database Authentication 32](#_Toc115190401)

[Kerberos Authentication 34](#_Toc115190401)

[Limitations 36](#_Toc115190413)

**Data Encryption in Amazon Aurora**

Document Name: Data Encryption in Amazon Aurora

Owner: Hema Paul

**Scope**

Aurora gives you the performance and availability of commercial-grade databases at one-tenth the cost. Amazon Aurora is a relational database management system (RDBMS) built for the cloud with full MySQL and PostgreSQL compatibility.

**Problem Statement**

Amazon Aurora encrypted DB clusters provide an additional layer of data protection by securing your data from unauthorized access to the underlying storage.

**Introduction**

Aurora is part of the managed database service Amazon Relational Database Service (Amazon RDS). Amazon RDS is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. An Amazon Aurora DB cluster consists of one or more DB instances and a cluster volume that manages the data for those DB instances. An Aurora cluster volume is a virtual database storage volume that spans multiple Availability Zones, with each Availability Zone having a copy of the DB cluster data.

**Data Encryption**

Nowadays encryption of data at rest is must, especially if it is stored somewhere in public cloud. It is necessary to meet various compliance requirements.

Keeping data and applications safe in the cloud is one of the most visible challenges facing cloud teams. Cloud storage services where data resides are frequently a target for hackers, not because the services are inherently weak but because they are often improperly configured.

# Encrypting Amazon RDS resources

Amazon Aurora encrypted DB clusters use the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your Amazon Aurora DB clusters. After your data is encrypted, Amazon Aurora handles authentication of access and decryption of your data transparently with a minimal impact on performance. You don't need to modify your database client applications to use encryption.

**How RDS Encryption Works**

With RDS-encrypted resources, **data is encrypted at rest, including the underlying storage for a database (DB) instance, its automated backups, read replicas, and snapshots**. This capability uses the open standard AES-256 encryption algorithm to encrypt your data, which is transparent to your database engine.

## **Overview of encrypting Amazon Aurora resources**

Amazon Aurora encrypted DB clusters provide an additional layer of data protection by securing your data from unauthorized access to the underlying storage. You can use Amazon Aurora encryption to increase data protection of your applications deployed in the cloud, and to fulfill compliance requirements for encryption at rest.

For an Amazon Aurora encrypted DB cluster, all DB instances, logs, backups, and snapshots are encrypted. You can also encrypt a read replica of an Amazon Aurora encrypted cluster. Amazon Aurora uses an AWS KMS key to encrypt these resources. Each DB instance in the DB cluster is encrypted using the same KMS key as the DB cluster. If you copy an encrypted snapshot, you can use a different KMS key to encrypt the target snapshot than the one that was used to encrypt the source snapshot.

You can use an AWS managed key, or you can create customer managed keys. To manage the customer managed keys used for encrypting and decrypting your Amazon Aurora resources, you use the [AWS Key Management Service (AWS KMS)](https://docs.aws.amazon.com/kms/latest/developerguide/). AWS KMS combines secure, highly available hardware and software to provide a key management system scaled for the cloud. Using AWS KMS, you can create customer managed keys and define the policies that control how these customer managed keys can be used. AWS KMS supports CloudTrail, so you can audit KMS key usage to verify that customer managed keys are being used appropriately. You can use your customer managed keys with Amazon Aurora and supported AWS services such as Amazon S3, Amazon EBS, and Amazon Redshift.

## **Encrypting a DB instance**

To encrypt a new DB instance, choose **Enable encryption** on the Amazon RDS console. For information on creating a DB instance, see [Creating an Amazon RDS DB instance](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateDBInstance.html).

If you use the [create-db-instance](https://docs.aws.amazon.com/cli/latest/reference/rds/create-db-instance.html) AWS CLI command to create an encrypted DB instance, set the --storage-encrypted parameter. If you use the [CreateDBInstance](https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_CreateDBInstance.html) API operation, set the StorageEncrypted parameter to true.

When you create an encrypted DB instance, you can choose a customer managed key or the AWS managed key for Amazon RDS to encrypt your DB instance. If you don't specify the key identifier for a customer managed key, Amazon RDS uses the AWS managed key for your new DB instance. Amazon RDS creates an AWS managed key for Amazon RDS for your AWS account. Your AWS account has a different AWS managed key for Amazon RDS for each AWS Region.

Once you have created an encrypted DB instance, you can't change the KMS key used by that DB instance. Therefore, be sure to determine your KMS key requirements before you create your encrypted DB instance.

If you use the AWS CLI create-db-instance command to create an encrypted DB instance with a customer managed key, set the --kms-key-id parameter to any key identifier for the KMS key. If you use the Amazon RDS API CreateDBInstance operation, set the KmsKeyId parameter to any key identifier for the KMS key. To use a customer managed key in a different AWS account, specify the key ARN or alias ARN.

**How to encrypt data in AWS RDS at rest?**

In the Amazon RDS console navigation pane, choose Snapshots, and select the DB snapshot you created. For Actions, choose Copy Snapshot. Provide the destination AWS Region and the name of the DB snapshot copy in the corresponding fields. Select the Enable Encryption checkbox.

**How to encrypt data in transit for AWS RDS?**

Simply **add the native network encryption option to an option group and associate that option group with the DB instance**. Once an encrypted connection is established, data transferred between the DB Instance and your application will be encrypted during transfer.

## **Determining whether encryption is turned on for a DB cluster**

You can use the AWS Management Console, AWS CLI, or RDS API to determine whether encryption at rest is turned on for a DB cluster.

**Console**

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>
2. In the navigation pane, choose **Databases**.
3. Choose the name of the DB cluster that you want to check to view its details.
4. Choose the **Configuration** tab and check the **Encryption** value.

It shows either **Enabled** or **Not enabled**.


                                    Checking encryption at rest for a DB cluster
                                

**AWS CLI**

To determine whether encryption at rest is turned on for a DB cluster by using the AWS CLI, call the [*describe-db-clusters*](https://docs.aws.amazon.com/cli/latest/reference/rds/describe-db-clusters.html) command with the following option:

* --db-cluster-identifier – The name of the DB cluster.

The following example uses a query to return either TRUE or FALSE regarding encryption at rest for the mydb DB cluster.

**Example**

aws rds describe-db-clusters --db-cluster-identifier *mydb* --query "\*[].{StorageEncrypted:StorageEncrypted}" --output

## **RDS API**

To determine whether encryption at rest is turned on for a DB cluster by using the Amazon RDS API, call the [*DescribeDBClusters*](https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_DescribeDBClusters.html) operation with the following parameter:

* DBClusterIdentifier – The name of the DB cluster.

## **Availability of Amazon Aurora encryption**

Amazon Aurora encryption is currently available for all database engines and storage types.

**Note:** Amazon Aurora encryption is **not** available for the db.t2.micro DB instance class.

## **Encrypting an Amazon Aurora DB cluster**

To encrypt a new DB cluster, choose **Enable encryption** on the console. For information on creating a DB cluster, see [Creating an Amazon Aurora DB cluster](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.CreateInstance.html).

If you use the [***create-db-cluster***](https://docs.aws.amazon.com/cli/latest/reference/rds/create-db-cluster.html) AWS CLI command to create an encrypted DB cluster, set the --storage-encrypted parameter. If you use the [CreateDBCluster](https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_CreateDBCluster.html) API operation, set the StorageEncrypted parameter to **true**.

When you create an encrypted DB cluster, you can choose a customer managed key or the AWS managed key for Amazon Aurora to encrypt your DB cluster. If you don't specify the key identifier for a customer managed key, Amazon Aurora uses the AWS managed key for your new DB cluster. Amazon Aurora creates an AWS managed key for Amazon Aurora for your AWS account. Your AWS account has a different AWS managed key for Amazon Aurora for each AWS Region.

Once you have created an encrypted DB cluster, you can't change the KMS key used by that DB cluster. Therefore, be sure to determine your KMS key requirements before you create your encrypted DB cluster.

If you use the AWS CLI create-db-cluster command to create an encrypted DB cluster with a customer managed key, set the --kms-key-id parameter to any key identifier for the KMS key. If you use the Amazon RDS API CreateDBInstance operation, set the KmsKeyId parameter to any key identifier for the KMS key. To use a customer managed key in a different AWS account, specify the key ARN or alias ARN.

**Important**

Amazon Aurora can lose access to the KMS key for a DB cluster. For example, Aurora loses access when the KMS key isn't enabled, or when Aurora access to a KMS key is revoked. In these cases, the encrypted DB cluster goes into *inaccessible-encryption-credentials-recoverable* state. The DB cluster remains in this state for seven days. When you start the DB cluster during that time, it checks if the KMS key is active and recovers the DB cluster if it is. Restart the DB cluster using the AWS CLI command [start-db-cluster](https://docs.aws.amazon.com/cli/latest/reference/rds/start-db-cluster.html). Currently, you can't start a DB cluster in this state using the AWS Management Console.

If the DB cluster isn't recovered, then it goes into the **terminal** *inaccessible-encryption-credentials* state. In this case, you can only restore the DB cluster from a backup. We strongly recommend that you always turn on backups for encrypted DB instances to guard against the loss of encrypted data in your databases.

**Types of Encryption used in Aurora**

**AES – 256 Encryption Algorithm**

AES-256, which has a key length of 256 bits, **supports the largest bit size and is practically unbreakable by brute force based on current computing power**, making it the strongest encryption standard. The AES Encryption algorithm (also known as the Rijndael algorithm) is **a symmetric block cipher algorithm with a block/chunk size of 128 bits**. It converts these individual blocks using keys of 128, 192, and 256 bits. Once it encrypts these blocks, it joins them together to form the ciphertext.

## **Envelope encryption**

When you encrypt your data, your data is protected, but you have to protect your encryption key. One strategy is to encrypt it. Envelope encryption is the practice of encrypting plaintext data with a data key, and then encrypting the data key under another key.

You can even encrypt the data encryption key under another encryption key, and encrypt that encryption key under another encryption key. But, eventually, one key must remain in plaintext so you can decrypt the keys and your data. This top-level plaintext key encryption key is known as the root key.


        Envelope encryption
      

AWS KMS helps you to protect your encryption keys by storing and managing them securely. Root keys stored in AWS KMS, known as [AWS KMS keys](https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#kms_keys), never leave the AWS KMS [FIPS validated hardware security modules](https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4177) unencrypted. To use a KMS key, you must call AWS KMS.


        Envelope encryption with multiple key encryption keys
      

Envelope encryption offers several benefits:

* **Protecting data keys**

When you encrypt a data key, you don't have to worry about storing the encrypted data key, because the data key is inherently protected by encryption. You can safely store the encrypted data key alongside the encrypted data.

* **Encrypting the same data under multiple keys**

Encryption operations can be time consuming, particularly when the data being encrypted are large objects. Instead of re-encrypting raw data multiple times with different keys, you can re-encrypt only the data keys that protect the raw data.

* **Combining the strengths of multiple algorithms**

In general, symmetric key algorithms are faster and produce smaller ciphertexts than public key algorithms. But public key algorithms provide inherent separation of roles and easier key management. Envelope encryption lets you combine the strengths of each strategy.

# AWS KMS key management

Amazon Aurora automatically integrates with AWS Key Management Service (AWS KMS) for key management. Amazon Aurora uses envelope encryption.

An AWS KMS key is a logical representation of a key. The KMS key includes metadata, such as the key ID, creation date, description, and key state. The KMS key also contains the key material used to encrypt and decrypt data.

You can manage KMS keys used for Amazon Aurora encrypted DB clusters using the [AWS Key Management Service (AWS KMS)](https://docs.aws.amazon.com/kms/latest/developerguide/) in the [AWS KMS console](https://console.aws.amazon.com/kms), the AWS CLI, or the AWS KMS API. If you want full control over a KMS key, then you must create a *customer managed key.*

AWS managed keys are KMS keys in your account that are created, managed, and used on your behalf by an AWS service that is integrated with AWS KMS. You can't delete, edit, or rotate AWS managed keys.You can't share a snapshot that has been encrypted using the AWS managed key of the AWS account that shared the snapshot.You can view audit logs of every action taken with an AWS managed or customer managed key by using [AWS CloudTrail](https://docs.aws.amazon.com/awscloudtrail/latest/userguide/).

**Important**

If you turn off or revoke permissions to a KMS key used by an RDS database, RDS puts your database into a terminal state when access to the KMS key is required. This change could be immediate, or deferred, depending on the use case that required access to the KMS key. In this state, the DB cluster is no longer available, and the current state of the database can't be recovered. To restore the DB cluster, you must re-enable access to the KMS key for RDS, and then restore the DB cluster from the latest available backup.

## **Authorizing use of a customer managed key**

When Aurora uses a customer managed key in cryptographic operations, it acts on behalf of the user who is creating or changing the Aurora resource.

To create an Aurora resource using a customer managed key, a user must have permissions to call the following operations on the customer managed key:

* kms:CreateGrant
* kms:DescribeKey

You can specify these required permissions in a key policy, or in an IAM policy if the key policy allows it.

You can make the IAM policy stricter in various ways. For example, to allow the customer managed key to be used only for requests that originate in Aurora, you can use the [kms: ViaService condition key](https://docs.aws.amazon.com/kms/latest/developerguide/policy-conditions.html#conditions-kms-via-service) with the rds.*<region*>.amazonaws.com value.

You can also use the keys or values in the [encryption context](https://docs.aws.amazon.com/kms/latest/developerguide/services-rds.html#rds-encryptioncontext) as a condition for using the customer managed key for cryptographic operations.

**Using SSL/TLS to encrypt a connection to a DB cluster**

You can use Secure Socket Layer (SSL) or Transport Layer Security (TLS) from your application to encrypt a connection to a DB cluster running Aurora MySQL or Aurora PostgreSQL.

SSL/TLS connections provide one layer of security by encrypting data that moves between your client and a DB cluster. Using a server certificate provides an extra layer of security by validating that the connection is being made to an Amazon Aurora DB cluster. It does so by checking the server certificate that is automatically installed on all DB clusters that you provision.

Each DB engine has its own process for implementing SSL/TLS. To learn how to implement SSL/TLS for your DB cluster, use the link following that corresponds to your DB engine:

* [Security with Amazon Aurora MySQL](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Security.html)
* [Security with Amazon Aurora PostgreSQL](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraPostgreSQL.Security.html)

**Security with Amazon Aurora MySQL**

Security for Amazon Aurora MySQL is managed at three levels:

* To control who can perform Amazon RDS management actions on Aurora MySQL DB clusters and DB instances, you use AWS Identity and Access Management (IAM).
* Make sure to create Aurora MySQL DB clusters in a virtual public cloud (VPC) based on the Amazon VPC service.
* To authenticate login and permissions for an Amazon Aurora MySQL DB cluster, you can take either of the following approaches, or a combination of them:
  + You can take the same approach as with a standalone instance of MySQL.
  + Commands such as CREATE USER, RENAME USER, GRANT, REVOKE, and SET PASSWORD work just as they do in on-premises databases, as does directly modifying database schema tables.
  + You can also use IAM database authentication.

## **Master user privileges with Amazon Aurora MySQL**

When you create an Amazon Aurora MySQL DB instance, the master user has the following default privileges:

* ALTER
* ALTER ROUTINE
* CREATE
* CREATE ROUTINE
* CREATE TEMPORARY TABLES
* CREATE USER
* CREATE VIEW
* DELETE
* DROP
* EVENT
* EXECUTE
* GRANT OPTION
* INDEX
* INSERT
* LOAD FROM S3
* LOCK TABLES
* PROCESS
* REFERENCES
* RELOAD
* REPLICATION CLIENT
* REPLICATION SLAVE
* SELECT
* SHOW DATABASES
* SHOW VIEW
* TRIGGER
* UPDATE

To provide management services for each DB cluster, the rdsadmin user is created when the DB cluster is created. Attempting to drop, rename, change the password, or change privileges for the rdsadmin account results in an error.

For management of the Aurora MySQL DB cluster, the standard kill and kill\_query commands have been restricted. Instead, use the Amazon RDS commands rds\_kill and rds\_kill\_query to terminate user sessions or queries on Aurora MySQL DB instances.

### **Encrypting connections to an Aurora MySQL DB cluster**

To encrypt connections using the default mysql client, launch the mysql client using the --ssl-ca parameter to reference the public key, for example:

For MySQL 5.7 and 8.0:

mysql -h myinstance.123456789012.rds-us-east-1.amazonaws.com

--ssl-ca=full\_path\_to\_CA\_certificate --ssl-mode=VERIFY\_IDENTITY

For MySQL 5.6:

mysql -h myinstance.123456789012.rds-us-east-1.amazonaws.com

--ssl-ca=full\_path\_to\_CA\_certificate --ssl-verify-server-cert

Replace full\_path\_to\_CA\_certificate with the full path to your Certificate Authority (CA) certificate.

You can require SSL/TLS connections for specific users accounts. For example, you can use one of the following statements, depending on your MySQL version, to require SSL/TLS connections on the user account encrypted\_user.

For MySQL 5.7 and 8.0:

ALTER USER 'encrypted\_user'@'%' REQUIRE SSL;

For MySQL 5.6:

GRANT USAGE ON \*.\* TO 'encrypted\_user'@'%' REQUIRE SSL;

When you use an RDS proxy, you connect to the proxy endpoint instead of the usual cluster endpoint. You can make SSL/TLS required or optional for connections to the proxy, in the same way as for connections directly to the Aurora DB cluster.

### **Configuring cipher suites for connections to Aurora MySQL DB clusters**

By using configurable cipher suites, you can have more control over the security of your database connections. You can specify a list of cipher suites that you want to allow to secure client SSL/TLS connections to your database. With configurable cipher suites, you can control the connection encryption that your database server accepts. Doing this prevents the use of insecure or deprecated ciphers.

For the client application, you can specify the ciphers to use for encrypted connections by using the --ssl-cipher option when connecting to the database.

You can also use the [describe-engine-default-cluster-parameters](https://docs.aws.amazon.com/cli/latest/reference/rds/describe-engine-default-cluster-parameters.html) CLI command to determine which cipher suites are currently supported for a specific parameter group family. The following example shows how to get the allowed values for the ssl\_cipher cluster parameter for Aurora MySQL 5.7.

*aws rds describe-engine-default-cluster-parameters --db-parameter-group-family aurora-mysql5.7*

...some output truncated...

{

"ParameterName": "ssl\_cipher",

"ParameterValue": "DHE-RSA-AES128-SHA,DHE-RSA-AES128-SHA256,DHE-RSA-AES128-GCM-SHA256,DHE-RSA-AES256-SHA,DHE-RSA-AES256-SHA256,DHE-RSA-AES256-GCM-SHA384,ECDHE-RSA-AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-AES128-GCM-SHA256,ECDHE-RSA-AES256-SHA,ECDHE-RSA-AES256-SHA384,ECDHE-RSA-AES256-GCM-SHA384",

"Description": "The list of permissible ciphers for connection encryption.",

"Source": "system",

"ApplyType": "static",

"DataType": "list",

"AllowedValues": "DHE-RSA-AES128-SHA,DHE-RSA-AES128-SHA256,DHE-RSA-AES128-GCM-SHA256,DHE-RSA-AES256-SHA,DHE-RSA-AES256-SHA256,DHE-RSA-AES256-GCM-SHA384,ECDHE-RSA-AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-AES128-GCM-SHA256,ECDHE-RSA-AES256-SHA,ECDHE-RSA-AES256-SHA384,ECDHE-RSA-AES256-GCM-SHA384",

"IsModifiable": true,

"SupportedEngineModes": [

"provisioned"

]

},

...some output truncated...

# Security with Amazon Aurora PostgreSQL

## **Securing Aurora PostgreSQL data with SSL/TLS**

Amazon RDS supports Secure Socket Layer (SSL) and Transport Layer Security (TLS) encryption for Aurora PostgreSQL DB clusters. Using SSL/TLS, you can encrypt a connection between your applications and your Aurora PostgreSQL DB clusters. You can also force all connections to your Aurora PostgreSQL DB cluster to use SSL/TLS. Amazon Aurora PostgreSQL supports Transport Layer Security (TLS) versions 1.1 and 1.2. We recommend using TLS 1.2 for encrypted connections.

SSL/TLS support is available in all AWS Regions for Aurora PostgreSQL. Amazon RDS creates an SSL/TLS certificate for your Aurora PostgreSQL DB cluster when the DB cluster is created. If you enable SSL/TLS certificate verification, then the SSL/TLS certificate includes the DB cluster endpoint as the Common Name (CN) for the SSL/TLS certificate to guard against spoofing attacks.

**To connect to an Aurora PostgreSQL DB cluster over SSL/TLS**

1. Download the certificate.
2. Import the certificate into your operating system.
3. Connect to your Aurora PostgreSQL DB cluster over SSL/TLS.

When you connect using SSL/TLS, your client can choose to verify the certificate chain or not. If your connection parameters specify sslmode=verify-ca or sslmode=verify-full, then your client requires the RDS CA certificates to be in their trust store or referenced in the connection URL. This requirement is to verify the certificate chain that signs your database certificate.

When a client, such as psql or JDBC, is configured with SSL/TLS support, the client first tries to connect to the database with SSL/TLS by default. If the client can't connect with SSL/TLS, it reverts to connecting without SSL/TLS. The default sslmode mode used is different between libpq-based clients (such as psql) and JDBC. The libpq-based clients default to prefer, where JDBC clients default to verify-full.

Use the sslrootcert parameter to reference the certificate, for example sslrootcert=rds-ssl-ca-cert.pem.

$ psql -h testpg.cdhmuqifdpib.us-east-1.rds.amazonaws.com -p 5432 \

"dbname=testpg user=testuser sslrootcert=rds-ca-2015-root.pem sslmode=verify-full"

### **Configuring cipher suites for connections to Aurora PostgreSQL DB clusters**

By using configurable cipher suites, you can have more control over the security of your database connections. You can specify a list of cipher suites that you want to allow to secure client SSL/TLS connections to your database. With configurable cipher suites, you can control the connection encryption that your database server accepts. Doing this helps prevent the use of insecure or deprecated ciphers.

Configurable cipher suites is supported in Aurora PostgreSQL versions 11.8 and higher.

To specify the list of permissible ciphers for encrypting connections, modify the ssl\_ciphers cluster parameter. Set the ssl\_ciphers parameter in a cluster parameter group using the AWS Management Console, the AWS CLI, or the RDS API. To set cluster parameters, see [Modifying parameters in a DB cluster parameter group](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/USER_WorkingWithDBClusterParamGroups.html#USER_WorkingWithParamGroups.ModifyingCluster).

Set the ssl\_ciphers parameter to a string of comma-separated cipher values. The valid ciphers include the following:

* ECDHE-RSA-AES128-SHA
* ECDHE-RSA-AES128-SHA256
* ECDHE-RSA-AES128-GCM-SHA256
* ECDHE-RSA-AES256-SHA
* ECDHE-RSA-AES256-GCM-SHA384

You can also use the [describe-engine-default-cluster-parameters](https://docs.aws.amazon.com/cli/latest/reference/rds/describe-engine-default-cluster-parameters.html) CLI command to determine which cipher suites are currently supported for a specific parameter group family. The following example shows how to get the allowed values for the ssl\_cipher cluster parameter for Aurora PostgreSQL 11.

aws rds describe-engine-default-cluster-parameters --db-parameter-group-family aurora-postgresql11

...some output truncated...

{

"ParameterName": "ssl\_ciphers",

"Description": "Sets the list of allowed TLS ciphers to be used on secure connections.",

"Source": "engine-default",

"ApplyType": "dynamic",

"DataType": "list",

"AllowedValues": "ECDHE-RSA-AES256-GCM-SHA384,ECDHE-RSA-AES256-SHA384,AES256-SHA,AES128-SHA,DES-CBC3-SHA,ADH-DES-CBC3-SHA,EDH-RSA-DES-CBC3-SHA,EDH-DSS-DES-CBC3-SHA,ADH-AES256-SHA,DHE-RSA-AES256-SHA,DHE-DSS-AES256-SHA,ADH-AES128-SHA,DHE-RSA-AES128-SHA,DHE-DSS-AES128-SHA,HIGH",

"IsModifiable": true,

"MinimumEngineVersion": "11.8",

"SupportedEngineModes": [

"provisioned"

]

},

...some output truncated...

**Note**

All certificates are only available for download using SSL/TLS connections.

To get a certificate bundle that contains both the intermediate and root certificates for all AWS Regions, download from <https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem>

If your application is on Microsoft Windows and requires a PKCS7 file, you can download the PKCS7 certificate bundle. This bundle contains both the intermediate and root certificates at <https://truststore.pki.rds.amazonaws.com/global/global-bundle.p7b>

**Note**

Amazon RDS Proxy and Aurora Serverless use certificates from the AWS Certificate Manager (ACM). If you are using RDS Proxy, you don't need to download Amazon RDS certificates or update applications that use RDS Proxy connections. For more information about using TLS/SSL with RDS Proxy, see [Using TLS/SSL with RDS Proxy](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/rds-proxy.howitworks.html#rds-proxy-security.tls).

If you are Aurora Serverless, downloading Amazon RDS certificates isn't required. For more information about using TLS/SSL with Aurora Serverless, see [Using TLS/SSL with Aurora Serverless v1](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-serverless.html#aurora-serverless.tls).

## **Certificate bundles for AWS Regions**

To get a certificate bundle that contains both the intermediate and root certificates for an AWS Region, download from the link for the AWS Region as in the given reference page.

*https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/UsingWithRDS.SSL.html*

## **AWS GovCloud (US) certificates**

To get a certificate bundle that contains both the intermediate and root certificates for the AWS GovCloud (US) Regions, download from [*https://truststore.pki.us-gov-west-1.rds.amazonaws.com/global/global-bundle.pem*](https://truststore.pki.us-gov-west-1.rds.amazonaws.com/global/global-bundle.pem)

If your application is on Microsoft Windows and requires a PKCS7 file, you can download the PKCS7 certificate bundle. This bundle contains both the intermediate and root certificates at [*https://truststore.pki.us-gov-west-1.rds.amazonaws.com/global/global-bundle.p7b*](https://truststore.pki.us-gov-west-1.rds.amazonaws.com/global/global-bundle.p7b)

To get a certificate bundle that contains both the intermediate and root certificates for an AWS GovCloud (US) Region, download from the link for the AWS GovCloud (US) Region as in the following reference page.

https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/UsingWithRDS.SSL.html

**Rotating your SSL/TLS certificate**

As of March 5, 2020, Amazon RDS CA-2015 certificates have expired. If you use or plan to use Secure Sockets Layer (SSL) or Transport Layer Security (TLS) with certificate verification to connect to your RDS DB instances, you require Amazon RDS CA-2019 certificates, which are enabled by default for new DB instances. If you currently do not use SSL/TLS with certificate verification, you might still have expired CA-2015 certificates and must update them to CA-2019 certificates if you plan to use SSL/TLS with certificate verification to connect to your RDS databases.

Follow these instructions to complete your updates. Before you update your DB instances to use the new CA certificate, make sure that you update your clients or applications connecting to your RDS databases.

Amazon RDS provides new CA certificates as an AWS security best practice. For information about the new certificates and the supported AWS Regions, see [Using SSL/TLS to encrypt a connection to a DB cluster](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/UsingWithRDS.SSL.html).

**Note**

Amazon RDS Proxy and Aurora Serverless use certificates from the AWS Certificate Manager (ACM). If you are using RDS Proxy, when you rotate your SSL/TLS certificate, you don't need to update applications that use RDS Proxy connections. For more information about using TLS/SSL with RDS Proxy, see [Using TLS/SSL with RDS Proxy](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/rds-proxy.howitworks.html#rds-proxy-security.tls).

If you are Aurora Serverless, rotating your SSL/TLS certificate isn't required. For more information about using TLS/SSL with Aurora Serverless, see [Using TLS/SSL with Aurora Serverless v1](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-serverless.html#aurora-serverless.tls)

**Note**

If you are using a Go version 1.15 application with a DB instance that was created or updated to the rds-ca-2019 certificate prior to July 28, 2020, you must update the certificate again. Run the modify-db-instance command shown in the AWS CLI section using rds-ca-2019 as the CA certificate identifier. In this case, it isn't possible to update the certificate using the AWS Management Console. If you created your DB instance or updated its certificate after July 28, 2020, no action is required.

## **Updating your CA certificate by modifying your DB instance**

Complete the following steps to update your CA certificate.

**To update your CA certificate by modifying your DB instance**

1. Download the new SSL/TLS certificate as described in [Using SSL/TLS to encrypt a connection to a DB cluster](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/UsingWithRDS.SSL.html).
2. Update your applications to use the new SSL/TLS certificate.

The methods for updating applications for new SSL/TLS certificates depend on your specific applications. Work with your application developers to update the SSL/TLS certificates for your applications.

For information about checking for SSL/TLS connections and updating applications for each DB engine, see the following topics:

* + [Updating applications to connect to Aurora MySQL DB clusters using new SSL/TLS certificates](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/ssl-certificate-rotation-aurora-mysql.html)
  + [Updating applications to connect to Aurora PostgreSQL DB clusters using new SSL/TLS certificates](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/ssl-certificate-rotation-aurora-postgresql.html)

**Note**

The certificate bundle contains certificates for both the old and new CA, so you can upgrade your application safely and maintain connectivity during the transition period. If you are using the AWS Database Migration Service to migrate a database to a DB cluster, we recommend using the certificate bundle to ensure connectivity during the migration.

1. Modify the DB instance to change the CA from **rds-ca-2015** to **rds-ca-2019**.

**Important**

By default, this operation restarts your DB instance. If you don't want to restart your DB instance during this operation, you can use the modify-db-instance CLI command and specify the --no-certificate-rotation-restart option.

This option will not rotate the certificate until the next time the database restarts, either for planned or unplanned maintenance. This option is only recommended if you don't use SSL/TLS.

If you are experiencing connectivity issues after certificate expiry, use the apply immediately option by specifying **Apply immediately** in the console or by specifying the --apply-immediately option using the AWS CLI. By default, this operation is scheduled to run during your next maintenance window.

You can use the AWS Management Console or the AWS CLI to change the CA certificate from **rds-ca-2015** to **rds-ca-2019** for a DB instance.

## **Console**

**To change the CA from rds-ca-2015 to rds-ca-2019 for a DB instance**

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to modify.
3. Choose **Modify**.


                                        Modify DB instance
                                    

The **Modify DB Instance** page appears.

1. In the **Connectivity** section, choose **rds-ca-2019**.


                                        Choose CA certificate
                                    

1. Choose **Continue** and check the summary of modifications.
2. To apply the changes immediately, choose **Apply immediately**.

**Important** :Choosing this option restarts your database immediately.

1. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

**Important** : When you schedule this operation, make sure that you have updated your client-side trust store beforehand.

Or choose **Back** to edit your changes or **Cancel** to cancel your changes.

## **AWS CLI**

To use the AWS CLI to change the CA from **rds-ca-2015** to **rds-ca-2019** for a DB instance, call the [modify-db-instance](https://docs.aws.amazon.com/cli/latest/reference/rds/modify-db-instance.html) command. Specify the DB instance identifier and the --ca-certificate-identifier option.

**Important**

When you schedule this operation, make sure that you have updated your client-side trust store beforehand.

**Example**

The following code modifies mydbinstance by setting the CA certificate to rds-ca-2019. The changes are applied during the next maintenance window by using --no-apply-immediately. Use --apply-immediately to apply the changes immediately.

**Important**

By default, this operation reboots your DB instance. If you don't want to reboot your DB instance during this operation, you can use the modify-db-instance CLI command and specify the --no-certificate-rotation-restart option.

This option will not rotate the certificate until the next time the database restarts, either for planned or unplanned maintenance. This option is only recommended if you do not use SSL/TLS.

Use --apply-immediately to apply the update immediately. By default, this operation is scheduled to run during your next maintenance window.

**For Linux, macOS, or Unix:**

aws rds modify-db-instance \

--db-instance-identifier mydbinstance \

--ca-certificate-identifier rds-ca-2019 \

--no-apply-immediately

**For Windows:**

aws rds modify-db-instance ^

--db-instance-identifier mydbinstance ^

--ca-certificate-identifier rds-ca-2019 ^

--no-apply-immediately

## **Updating your CA certificate by applying DB instance maintenance**

Complete the following steps to update your CA certificate by applying DB instance maintenance.

**To update your CA certificate by applying DB instance maintenance**

1. Download the new SSL/TLS certificate as described in [Using SSL/TLS to encrypt a connection to a DB cluster](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/UsingWithRDS.SSL.html).
2. Update your database applications to use the new SSL/TLS certificate.

The methods for updating applications for new SSL/TLS certificates depend on your specific applications. Work with your application developers to update the SSL/TLS certificates for your applications.

For information about checking for SSL/TLS connections and updating applications for each DB engine, see the following topics:

* + [Updating applications to connect to Aurora MySQL DB clusters using new SSL/TLS certificates](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/ssl-certificate-rotation-aurora-mysql.html)
  + [Updating applications to connect to Aurora PostgreSQL DB clusters using new SSL/TLS certificates](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/ssl-certificate-rotation-aurora-postgresql.html)

For a sample script that updates a trust store for a Linux operating system, see [Sample script for importing certificates into your trust store](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/UsingWithRDS.SSL-certificate-rotation.html#UsingWithRDS.SSL-certificate-rotation-sample-script).

**Note**

The certificate bundle contains certificates for both the old and new CA, so you can upgrade your application safely and maintain connectivity during the transition period.

1. Apply DB instance maintenance to change the CA from **rds-ca-2015** to **rds-ca-2019**.

**Important**

You can choose to apply the change immediately. By default, this operation is scheduled to run during your next maintenance window.

You can use the AWS Management Console to apply DB instance maintenance to change the CA certificate from **rds-ca-2015** to **rds-ca-2019** for multiple DB instances.

### **Updating your CA certificate by applying maintenance to multiple DB instances**

Use the AWS Management Console to change the CA certificate for multiple DB instances.

**To change the CA from rds-ca-2015 to rds-ca-2019 for multiple DB instances**

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>
2. In the navigation pane, choose **Databases**.
3. In the navigation pane, there is a **Certificate update** option that shows the total number of affected DB instances.


                                    Certificate rotation navigation pane option
                                

Choose **Certificate update** in the navigation pane.

The **Update your Amazon RDS SSL/TLS certificates** page appears.


                                    Update CA certificate for multiple DB instances
                                 **Note**

This page only shows the DB instances for the current AWS Region. If you have DB instances in more than one AWS Region, check this page in each AWS Region to see all DB instances with old SSL/TLS certificates.

1. Choose the DB instance you want to update.

You can schedule the certificate rotation for your next maintenance window by choosing **Update at the next maintenance window**. Apply the rotation immediately by choosing **Update now**.

**Important**

When your CA certificate is rotated, the operation restarts your DB instance.

If you experience connectivity issues after certificate expiry, use the **Update now** option.

1. If you choose **Update at the next maintenance window** or **Update now**, you are prompted to confirm the CA certificate rotation.

**Important**

Before scheduling the CA certificate rotation on your database, update any client applications that use SSL/TLS and the server certificate to connect. These updates are specific to your DB engine. To determine whether your applications use SSL/TLS and the server certificate to connect, see [Step 2: Update Your Database Applications to Use the New SSL/TLS Certificate](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/UsingWithRDS.SSL-certificate-rotation.html#UpdateAppsForSSL). After you have updated these client applications, you can confirm the CA certificate rotation.


                                    Confirm certificate rotation
                                

To continue, choose the check box, and then choose **Confirm**.

1. Repeat steps 3 and 4 for each DB instance that you want to update.

## **Sample script for importing certificates into your trust store**

The following are sample shell scripts that import the certificate bundle into a trust store.

Each sample shell script uses keytool, which is part of the Java Development Kit (JDK). For information about installing the JDK, see [JDK Installation Guide](https://docs.oracle.com/en/java/javase/17/install/overview-jdk-installation.html)

**Sample script for importing certificates on Linux**

The following is a sample shell script that imports the certificate bundle into a trust store on a Linux operating system.

mydir=tmp/certs

if [ ! -e "${mydir}" ]

then

mkdir -p "${mydir}"

fi

truststore=${mydir}/rds-truststore.jks

storepassword=changeit

curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem" > ${mydir}/global-bundle.pem

awk 'split\_after == 1 {n++;split\_after=0} /-----END CERTIFICATE-----/ {split\_after=1}{print > "rds-ca-" n ".pem"}' < ${mydir}/global-bundle.pem

for CERT in rds-ca-\*; do

alias=$(openssl x509 -noout -text -in $CERT | perl -ne 'next unless /Subject:/; s/.\*(CN=|CN = )//; print')

echo "Importing $alias"

keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -keystore ${truststore} -noprompt

rm $CERT

done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias | cut -d " " -f3- | while read alias

do

expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} -alias "${alias}" | grep Valid | perl -ne 'if(/until: (.\*?)\n/) { print "$1\n"; }'`

echo " Certificate ${alias} expires in '$expiry'"

done

**Automation of back ups**

Amazon RDS automatically creates a [backup](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_WorkingWithAutomatedBackups.html) of the underlying storage volumes associated with your Amazon RDS DB instance. All of your data is backed up into Amazon Simple Storage Service (Amazon S3), based on a retention policy that runs for up to 35 days. These backups allow you to perform a [**point-in-time recovery (PITR)**](https://docs.aws.amazon.com/aws-backup/latest/devguide/point-in-time-recovery.html) at any time within your retention period.

By default, this automation occurs only once each day during a [*30 minute backup window*](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_WorkingWithAutomatedBackups.html#USER_WorkingWithAutomatedBackups.BackupWindow)*.* If you need the automated backup to run more frequently, you can use the following methods:

* [Create an automated backup job](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateSnapshot.html) by AWS Lambda or a cron job on Amazon Elastic Compute Cloud (Amazon EC2). The automated backup job must call the **[CreateDBSnapshot](https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_CreateDBSnapshot.html" \t "_blank)** action at a required interval and a subsequent **[DeleteDBSnapshot](https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_DeleteDBSnapshot.html" \t "_blank)** action to automate the [deletion of snapshots](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_DeleteSnapshot.html).  
  --Or--
* Use [**AWS Backup**](https://aws.amazon.com/getting-started/hands-on/amazon-rds-backup-restore-using-aws-backup/), automating the scheduled backup.

### Create an automated backup job in Amazon RDS using AWS Backup

To create an automated backup job in Amazon RDS, which captures daily snapshots at a specific interval, perform the following steps:

1.    Open the [AWS Backup console](https://console.aws.amazon.com/backup/).

2.    Choose **Build a new plan** to create a new backup plan.

3.    Enter the **Backup plan name** and any relevant tag information.

4.    Under **Backup configuration**, update the following:

a. Backup plan name  
b. Backup vault  
c. Backup frequency  
d. Backup window

For example, if you want to schedule your automated backup to run every six hours, and to complete within three hours, update the following:  
For Backup frequency, select the **Custom cron expression** and type "cron(0 0/6 ? \* \* \*)" for your cron expression.  
For Backup window, select "Start within 1 hour" and "Complete within 3 hours".

5.    Choose **Copy automatically to a target region**.

**Note:** Not all AWS Regions are supported for automatic cross-Region copies. For more information, see[*Replicating automated backups to another AWS region.*](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ReplicateBackups.html)

6.    Specify the desired retention period for your automated backup. Amazon RDS Backup will retain the backup for the specified duration until the backup is automatically deleted.

7.    Choose **Create plan**.

8.    Under the **Newly created backup plan** tab, choose **Assign resources**.

9.    Assign your resources by updating the following information:  
Resource assignment name  
AWS Identity Access Management (IAM) role (to create and manage DB instance recovery points)  
Amazon RDS resource (by tag or resource ID)

### **Downtime for enabling Amazon RDS automated backups**

When you [enable Amazon RDS automated backups](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_WorkingWithAutomatedBackups.html#USER_WorkingWithAutomatedBackups.Enabling), an outage can occur when you update the backup retention period from "0" to a nonzero value. An outage can also occur when you update from a nonzero value to "0". The outage will be equivalent to the duration of a reboot and any engine recovery tasks performed during the engine startup.

**Note:** If you disable automated backups in Amazon RDS, all of your previous automated backup jobs will also be deleted.

### **How to get notified if AWS Backup job has failed?**

If your AWS Backup job has failed, configure your backup vault to send notifications to an SNS topic, notifying you of your job status. For more information, see [How can I get notifications for AWS Backup jobs that failed?](https://aws.amazon.com/premiumsupport/knowledge-center/aws-backup-failed-job-notification/)

### **What happens if my Amazon RDS automated backup fails?**

The following conditions must be met for Amazon RDS to successfully perform an automated backup:

* Your DB instance must be in the AVAILABLE state for automated backups to occur.
* A copy of your DB instance cannot be running in the same AWS Region as the original DB instance.

Although unlikely, if your Amazon RDS automated backup fails and the process does not complete before the end of a scheduled backup, Amazon RDS will retry the process. The automated backup will then take place during the next backup window. Otherwise, if your backup is progressing, it will (and can) continue beyond the specified window.

**Tip:** Use AWS Backup to manage both Amazon RDS snapshots and continuous backups with point-in-time recovery. Otherwise, your backups will fail when initiated during the default 30 minute backup window. To troubleshoot any issues with creating backups in Amazon RDS, see [Troubleshooting creating resources](https://docs.aws.amazon.com/aws-backup/latest/devguide/troubleshooting.html#troubleshooting-create-backup).

# Database authentication with Amazon RDS

Password, Kerberos, and IAM database authentication use different methods of authenticating to the database. Therefore, a specific user can log in to a database using only one authentication method.

## **Password Authentication**

With password authentication, your database performs all administration of user accounts. You create users with SQL statements such as CREATE USER, with the appropriate clause required by the DB engine for specifying passwords. For example, in MySQL the statement is CREATE USER name IDENTIFIED BY password, while in PostgreSQL, the statement is CREATE USER name WITH PASSWORD password.

With password authentication, your database controls and authenticates user accounts. If a DB engine has strong password management features, they can enhance security. Database authentication might be easier to administer using password authentication when you have small user communities. Because clear text passwords are generated in this case, integrating with AWS Secrets Manager can enhance security.

# Create an AWS Secrets Manager database secret

To store credentials for Amazon RDS, Amazon Aurora, Amazon Redshift, or Amazon DocumentDB, follow these steps. When you use the AWS CLI or one of the SDKs to store the secret, you must provide the secret in the [JSON structure of a database secret](https://docs.aws.amazon.com/secretsmanager/latest/userguide/reference_secret_json_structure.html). When you use the console to store a database secret, Secrets Manager automatically creates it in the correct JSON structure.

When you store database credentials for a source database that is replicated to other Regions, the secret contains connection information for the source database. If you then replicate the secret, the replicas are copies of the source secret and contain the same connection information. You can add additional key/value pairs to the secret for regional connection information.

To create a secret, you need the permissions granted by the **SecretsManagerReadWrite** [AWS managed policy](https://docs.aws.amazon.com/secretsmanager/latest/userguide/reference_available-policies.html).

**Steps :**

1. Open the Secrets Manager console at <https://console.aws.amazon.com/secretsmanager/>
2. Choose **Store a new secret**.
3. On the **Choose secret type** page, do the following:
   1. For **Secret type**, choose the type of database credentials to store:
      1. **Amazon RDS database** (includes Aurora)
      2. **Amazon DocumentDB database**
      3. **Amazon Redshift cluster**
   2. For **Credentials**, enter the credentials for the database.
   3. For **Encryption key**, choose the AWS KMS key that Secrets Manager uses to encrypt the secret value:
      1. For most cases, choose **aws/secretsmanager** to use the AWS managed key for Secrets Manager. There is no cost for using this key.
      2. If you need to access the secret from another AWS account, or if you want to use your own KMS key so that you can rotate it or apply a key policy to it, choose a customer managed key from the list or choose **Add new key** to create one. You will be charged for KMS keys that you create.
      3. You must have the following permissions to the key: kms:Encrypt, kms:Decrypt, and kms:GenerateDataKey. For more information about cross-account access, see [Permissions to AWS Secrets Manager secrets for users in a different account](https://docs.aws.amazon.com/secretsmanager/latest/userguide/auth-and-access_examples_cross.html)
   4. For **Database**, choose your database.
   5. Choose **Next**.
4. On the **Configure secret** page, do the following:
   1. Enter a descriptive **Secret name** and **Description**. Secret names must contain 1-512 Unicode characters.
   2. (Optional) In the **Tags** section, add tags to your secret. For tagging strategies, see [Tag AWS Secrets Manager secrets](https://docs.aws.amazon.com/secretsmanager/latest/userguide/managing-secrets_tagging.html). Don't store sensitive information in tags because they aren't encrypted.
   3. (Optional) In **Resource permissions**, to add a resource policy to your secret, choose **Edit permissions**. For more information, see [Attach a permissions policy to an AWS Secrets Manager secret](https://docs.aws.amazon.com/secretsmanager/latest/userguide/auth-and-access_resource-policies.html).
   4. (Optional) In **Replicate secret**, to replicate your secret to another AWS Region, choose **Replicate secret**. You can replicate your secret now or come back and replicate it later. For more information, see [Replicate a secret to other Regions](https://docs.aws.amazon.com/secretsmanager/latest/userguide/create-manage-multi-region-secrets.html).
   5. Choose **Next**.
5. (Optional) On the **Configure rotation** page, you can turn on automatic rotation. You can also keep rotation off for now and then turn it on later. For more information, see [Rotate secrets](https://docs.aws.amazon.com/secretsmanager/latest/userguide/rotating-secrets.html). Choose **Next**.
6. On the **Review** page, review your secret details, and then choose **Store**.

Secrets Manager returns to the list of secrets. If your new secret doesn't appear, choose the refresh button.

## **AWS CLI**

To create a secret by using the AWS CLI, first create a JSON file that contains your secret. For Secrets Manager to be able to rotate the secret, you must make sure the JSON matches the [JSON structure of a database secret](https://docs.aws.amazon.com/secretsmanager/latest/userguide/reference_secret_json_structure.html). For more information, see [Set up automatic rotation for Amazon RDS, Amazon Redshift, or Amazon DocumentDB secrets using the console](https://docs.aws.amazon.com/secretsmanager/latest/userguide/rotate-secrets_turn-on-for-db.html).

Then use the [create-secret](https://docs.aws.amazon.com/cli/latest/reference/secretsmanager/create-secret.html) operation to store the secret in Secrets Manager.

**To create a secret**

1. Create your secret in a file, for example a JSON file named mycreds.json.

{

"engine": "mysql",

"host": "<instance host name/resolvable DNS name>",

"username": "<username>",

"password": "<password>",

"dbname": "<database name. If not specified, defaults to None>",

"port": "<TCP port number. If not specified, defaults to 3306>"

}

1. In the AWS CLI, use the following command.

aws secretsmanager create-secret --name MySecret --secret-string file://mycreds.json

1. The following output is shown.

{

"SecretARN": "arn:aws:secretsmanager:Region:AccountId:secret:MySecret-a1b2c3",

"SecretName": "MySecret",

"SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE"

}

## **IAM Database Authentication**

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

An authentication token is a unique string of characters that Amazon RDS generates on request. Authentication tokens are generated using AWS Signature Version 4. Each token has a lifetime of 15 minutes. You don't need to store user credentials in the database, because authentication is managed externally using IAM. You can also still use standard database authentication. The token is only used for authentication and doesn't affect the session after it is established.

Generates an auth token used to connect to a db with IAM credentials.

generate-db-auth-token

--hostname <value>

--port <value>

--username <value>

[--debug]

[--endpoint-url <value>]

[--no-verify-ssl]

[--no-paginate]

[--output <value>]

[--query <value>]

[--profile <value>]

[--region <value>]

[--version <value>]

[--color <value>]

[--no-sign-request]

[--ca-bundle <value>]

[--cli-read-timeout <value>]

[--cli-connect-timeout <value>]

## **Kerberos Authentication**

Amazon RDS supports external authentication of database users using Kerberos and Microsoft Active Directory. Kerberos is a network authentication protocol that uses tickets and symmetric-key cryptography to eliminate the need to transmit passwords over the network. Kerberos has been built into Active Directory and is designed to authenticate users to network resources, such as databases.

Amazon RDS support for Kerberos and Active Directory provides the benefits of single sign-on and centralized authentication of database users. You can keep your user credentials in Active Directory. Active Directory provides a centralized place for storing and managing credentials for multiple DB instances.

You can make it possible for your database users to authenticate against DB instances in two ways. They can use credentials stored either in AWS Directory Service for Microsoft Active Directory or in your on-premises Active Directory.

Microsoft SQL Server, MySQL, and PostgreSQL DB instances support one- and two-way forest trust relationships. Oracle DB instances support one- and two-way external and forest trust relationships

## **Limitations of Amazon Aurora encrypted DB clusters**

* You can't turn off encryption on an encrypted DB cluster.
* You can't create an encrypted snapshot of an unencrypted DB cluster.
* A snapshot of an encrypted DB cluster must be encrypted using the same KMS key as the DB cluster.
* You can't convert an unencrypted DB cluster to an encrypted one. However, you can restore an unencrypted snapshot to an encrypted Aurora DB cluster. To do this, specify a KMS key when you restore from the unencrypted snapshot.
* You can't create an encrypted Aurora Replica from an unencrypted Aurora DB cluster. You can't create an unencrypted Aurora Replica from an encrypted Aurora DB cluster.
* To copy an encrypted snapshot from one AWS Region to another, you must specify the KMS key in the destination AWS Region. This is because KMS keys are specific to the AWS Region that they are created in.

The source snapshot remains encrypted throughout the copy process. Amazon Aurora uses ***envelope encryption*** to protect data during the copy process. For more information about envelope encryption, see [Envelope encryption](https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#enveloping) in the AWS Key Management Service Developer Guide.

* You can't unencrypt an encrypted DB cluster. However, you can export data from an encrypted DB cluster and import the data into an unencrypted DB cluster.