Data Encryption in RDS

Contents

[1. How to use different encryption key 3](#_Toc118189554)

[1.1 To create a copy of an RDS DB instance with a new encryption key: 3](#_Toc118189555)

[1.2 To delete a DB instance 3](#_Toc118189556)

[1.3 AWS CLI commands To delete a DB instance 4](#_Toc118189557)

[2. How to use custom encryption key 5](#_Toc118189558)

[2.1 Introducing AWS KMS 5](#_Toc118189559)

[2.2 Creating keys 6](#_Toc118189560)

[2.3 Creating an encrypted RDS database cluster 6](#_Toc118189561)

[2.4 Encrypted database snapshots 6](#_Toc118189562)

[2.4.1Creating RDS snapshots 6](#_Toc118189563)

[2.4.2 Copying encrypted RDS snapshots to other AWS Regions 6](#_Toc118189564)

[2.5 Rotating keys in AWS KMS 8](#_Toc118189565)

[2.6 Controlling AWS KMS key access 8](#_Toc118189566)

[3. Default encryption for EVERY Database 10](#_Toc118189567)

[4. Multi AZ for encryption 11](#_Toc118189568)

[AWS Encryption SDK message format reference 11](#_Toc118189569)

[5. Automation for this & EVRY topic. 12](#_Toc118189570)

[5.1 Best practices for automating DB instance creation 12](#_Toc118189571)

[5.2 Using the AWS CLI – create-db-instance 12](#_Toc118189572)

[5.2.1 Example 13](#_Toc118189573)

[5.3 Using AWS CloudFormation – AWS::RDS::DBInstance 13](#_Toc118189574)

[5.3.1Templates 13](#_Toc118189575)

[6. Encryption in transit 15](#_Toc118189576)

[6.1 Enforce encryption in transit with RDS Aurora PostgreSQL 15](#_Toc118189577)

[Create custom Aurora PostgreSQL parameter group 15](#_Toc118189578)

[6.2 Change the parameter group of the Aurora PostgreSQL database cluster 15](#_Toc118189579)

[6.3 Connect to your encrypted cluster using Cloud9 16](#_Toc118189580)

[7. Encryption at Rest – RDS 17](#_Toc118189581)

[8. To determine if your RDS database instances are encrypted 22](#_Toc118189582)

[8.1 Using AWS Console: 22](#_Toc118189583)

[8.2Using AWS CLI 23](#_Toc118189584)

[9. To enable data encryption for your existing RDS instances 24](#_Toc118189585)

[9.1 Using AWS Console 24](#_Toc118189586)

[9.2 Using AWS CLI 25](#_Toc118189587)

[10. Implementation 28](#_Toc118189588)

[11. Limitations of Amazon RDS encrypted DB instances 35](#_Toc118189588)

[12. Links: 36](#_Toc118189589)

# How to use different encryption key

You can't change the encryption key used by an Amazon RDS DB instance. However, you can create a copy of the RDS DB instance, and then choose a new encryption key for that copy.

Note: Data in [unlogged tables](https://www.postgresql.org/docs/current/sql-createtable.html) might not be restored using snapshots. For more information, review [Best practices for working with PostgreSQL](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_BestPractices.html#CHAP_BestPractices.PostgreSQL).

## 1.1 To create a copy of an RDS DB instance with a new encryption key:

You must do the following:

1. Open the [Amazon RDS console](https://console.aws.amazon.com/rds/).
2. In the navigation pane, choose Databases.
3. Choose the DB instance for which you want to create a manual snapshot.
4. [Create a manual snapshot](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateSnapshot.html) for your DB instance.
5. In the navigation pane, choose Snapshots.
6. Select the manual snapshot that you created.
7. Choose Actions, and then choose Copy Snapshot.
8. Under Encryption, select Enable Encryption.
9. For AWS KMS Key, choose the new encryption key that you want to use.
10. Choose Copy snapshot.
11. [Restore the copied snapshot](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_RestoreFromSnapshot.html).

The new RDS DB instance uses your new encryption key.

Confirm that your new database has all necessary data and your application is using the new database. When you no longer need the old RDS DB instance, you can [delete the instance](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_DeleteInstance.html).

## 1.2 To delete 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 delete.
3. For **Actions**, choose **Delete**.
4. To create a final DB snapshot for the DB instance, choose **Create final snapshot?**.
5. If you chose to create a final snapshot, enter the **Final snapshot name**.
6. To retain automated backups, choose **Retain automated backups**.
7. Enter **delete me** in the box.
8. Choose **Delete**.

## 1.3 AWS CLI commands To delete a DB instance

To delete a DB instance by using the AWS CLI, call the [delete-db-instance](https://docs.aws.amazon.com/cli/latest/reference/rds/delete-db-instance.html) command with the following options:

* --db-instance-identifier
* --final-db-snapshot-identifier or --skip-final-snapshot

**Example With a final snapshot and no retained automated backups**

For Linux, macOS, or Unix:

aws rds delete-db-instance \

--db-instance-identifier *mydbinstance* \

--final-db-snapshot-identifier *mydbinstancefinalsnapshot* \

--delete-automated-backups

For Windows:

aws rds delete-db-instance ^

--db-instance-identifier *mydbinstance* ^

--final-db-snapshot-identifier *mydbinstancefinalsnapshot* ^

--delete-automated-backups

**Example With retained automated backups and no final snapshot**

For Linux, macOS, or Unix:

aws rds delete-db-instance \

--db-instance-identifier *mydbinstance* \

--skip-final-snapshot \

--no-delete-automated-backups

For Windows:

aws rds delete-db-instance ^

--db-instance-identifier *mydbinstance* ^

--skip-final-snapshot ^

--no-delete-automated-backups

# How to use custom encryption key

[Amazon Relational Database Service (Amazon RDS)](https://aws.amazon.com/rds/) can encrypt data using an [AWS managed key or a Customer managed key (CMK)](https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#key-mgmt). However, some customers prefer to encrypt data using non-default keys. When using non-default keys, robust key creation, management, and deletion tools and processes are vital to data security. Such tools and processes also help prevent key expiration from affecting data availability.

## 2.1 Introducing AWS KMS

Key permissions fully integrate with [AWS Identity and Access Management (IAM)](https://aws.amazon.com/iam). Plus, all KMS API calls write to [AWS CloudTrail](https://aws.amazon.com/cloudtrail), providing a full audit trail of key creation, usage, and deletion. AWS KMS keys are 256 bit in length and use the Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM).

|  |  |  |  |
| --- | --- | --- | --- |
| MK type | Can view? | Can manage? | Is it used only for my AWS account? |
| AWS owned CMK | No | No | No |
| AWS managed CMK | Yes | No | Yes |
| Customer managed CMK | Yes | Yes | Yes |

[AWS owned keys](https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#aws-owned-cmk) are a collection of AWS KMS keys that an AWS service owns and manages for use in multiple AWS accounts.

AWS creates and fully manages [AWS managed keys](https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#aws-managed-cmk). [Customer managed keys](https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#customer-cmk) are CMKs in your AWS account that you create, own, and manage.

The source key material used to encrypt and decrypt data with a customer managed CMK can be provided by multiple sources, as shown in the following table.

|  |  |
| --- | --- |
| **Customer managed CMK type** | **Key material provided by** |
| Customer managed | AWS KMS |
| Customer managed external | You (using an external source, that is, HSM or OpenSSL) |
| Custom key store | You (using CloudHSM) |

## 2.2 Creating keys

Open the [AWS KMS console](https://console.aws.amazon.com/kms/home) and create a **Customer Managed Key**.

## 2.3 Creating an encrypted RDS database cluster

To encrypt a database, you must identify an AWS KMS key during database cluster creation. This key encrypts all data stored on the volume(s) used by RDS. After data is encrypted, it is inaccessible without AWS KMS key permissions. Attempts to restore an RDS snapshot or start a stopped RDS instance fail without that permission.

## 2.4 Encrypted database snapshots

All the database snapshots that you create from an encrypted RDS instance retain their encryption, sharing the same AWS KMS key. You can build a new database from such a snapshot, but only if you have access to the AWS KMS key used to encrypt the DB snapshot.

### 2.4.1Creating RDS snapshots

To create an RDS snapshot, select the database cluster instance, and choose **Actions**, **Take snapshot**.

### 2.4.2 Copying encrypted RDS snapshots to other AWS Regions

After you create RDS snapshots, you can copy encrypted RDS snapshots to other AWS Regions by following the steps described in this section. AWS KMS keys are Regional constructs. So, to copy a snapshot to another Region, you first must create an AWS KMS key in the destination Region. Use this new key to encrypt the snapshot in the destination Region.

1. First, choose your destination Region and create a key in that Region using the same procedure described earlier. Record the **key-id** and **key-alias** values for the newly created key.
2. Open the RDS console in the Region where the current RDS snapshot resides.
3. Choose **Snapshots** and select the RDS snapshot to copy. Choose **Action**, **Copy Snapshot**.
4. Select the destination Region and give the snapshot a friendly identifier. Copy any tags to align with your tagging strategy.
5. Under **Encryption**, select the new key in the destination Region, and choose **Copy Snapshot**.

Use the following bash script to copy database cluster snapshots to different Regions at regular intervals automatically. You might find this practice helpful for testing purposes or as part of a disaster recovery solution.

To use this script, copy and paste the text into a local file and modify the top-level variables to suit your environment. Also, be sure to modify the permissions of the local file to include the execute permission. Finally, configure your [AWS CLI](https://aws.amazon.com/cli/) with a suitable IAM user or role, then run the script.

#!/bin/bash

# Adjust these variables

DBINSTANCE=rds-demo-cluster

SRC\_AWS\_ACCOUNT=111111111111

SRC\_AWS\_REGION=eu-west-1

DEST\_AWS\_REGION=eu-central-1

DEST\_KMS\_KEYID=1111111-2222-3333-4444444444444444

# ----------------------------------------

# Don't change anything else after this

# ----------------------------------------

DATESR=$(date "+%Y%m%d%H%m")

# Dynamically generate snapshotId based on date

SNAPSHOTNAME=$DBINSTANCE-snapshot-$DATESR

# Generate snapshot Arn

SRCSNAPSHOTID=arn:aws:rds:$SRC\_AWS\_REGION:$SRC\_AWS\_ACCOUNT:cluster-snapshot:$SNAPSHOTNAME

echo "Creating snapshot for DB Instance: $DBINSTANCE"

aws rds create-db-cluster-snapshot \

--db-cluster-identifier $DBINSTANCE \

--db-cluster-snapshot-identifier $SNAPSHOTNAME \

--region $SRC\_AWS\_REGION

echo "Waiting for cluster snapshot $SNAPSHOTNAME to be available"

while STATUS=$(aws rds describe-db-cluster-snapshots --region $SRC\_AWS\_REGION --query "DBClusterSnapshots[?DBClusterSnapshotIdentifier=='$SNAPSHOTNAME'].Status" --output text); test "$STATUS" != "available"; do sleep 1 && echo "State: $STATUS"; done;

echo "State: $STATUS"

echo "Copying DB Snapshot...this may take a while depending on the size of the snapshot"

aws rds copy-db-cluster-snapshot \

--source-db-cluster-snapshot-identifier $SRCSNAPSHOTID \

--target-db-cluster-snapshot-identifier $SNAPSHOTNAME \

--source-region $SRC\_AWS\_REGION \

--kms-key-id $DEST\_KMS\_KEYID \

--region $DEST\_AWS\_REGION

## 2.5 Rotating keys in AWS KMS

Keys are commonly rotated to limit the impact of potential key compromise. If your CMKs come from original AWS KMS key material, you can opt to have AWS automatically rotate your CMK every year. You can opt for automatic annual rotation either when prompted during key creation, or after AWS KMS key creation. See [How to Enable and Disable Automatic Key Rotation](https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html#rotate-keys-console) for more information. For more detailed information about AWS KMS key rotation, see [Rotating Custom Master Keys](https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html).

If you obtained the original key material from your hardware security module (HSM) or another external source, you must [manually rotate your CMK](https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html). For RDS, you can create a new RDS instance or cluster from a re-encrypted snapshot copy using the following procedure:

1. Create a new CMK using the procedure outlined previously.
2. Create a snapshot of your existing instance or cluster using the procedure outlined previously. If you have a recent snapshot, you can skip this step.
3. Create a copy of the snapshot using the procedure described earlier in this post. However, you can also re-encrypt the new snapshot copy with the new CMK by selecting the new CMK in the **Encryption**
4. Select the new snapshot copy and choose **Actions**, **Restore Snapshot**.
5. Follow the online workflow to create a new RDS instance or cluster. The data is encrypted using the new CMK. For guidance, follow the new instance or cluster creation procedure described earlier.

## 2.6 Controlling AWS KMS key access

You manage access to encrypted resources in AWS by controlling access to the respective AWS KMS keys. Take a look at [Managing Access to AWS KMS CMKs](https://docs.aws.amazon.com/kms/latest/developerguide/control-access-overview.html#managing-access) for more information.

You can revoke access to an AWS KMS key, and thus revoke access to all AWS resources encrypted using that key, using any of the following methods:

* **Disable the AWS KMS key**. This procedure is reversible as the key can be later enabled.
* **Delete the AWS KMS key**. This procedure schedules the deletion of the key but doesn’t delete the key immediately. You specify a deletion period of between seven and thirty days. You can cancel the key deletion at any time within the specified deletion period. After that time, deletion is permanent. **Warning:** You cannot recover data encrypted using a deleted key.
* **Remove externally provided key material.** If you use externally provided key material, you can remove this key material. Removal effectively disables the key. This operation can be reversed by re-importing the key material. This option provides a high degree of confidence that the data cannot be accessed but also allows you to restore access at a later date.

# Default encryption for EVERY Database

On a database instance running with Amazon RDS encryption, data stored at rest in the underlying storage is encrypted, as are its automated backups, read replicas, and snapshots. Amazon RDS encryption uses the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your Amazon RDS instance.

# Multi AZ for encryption

Some customers deploy their database in a multi-region architecture because their applications [scale reads in a different Region](https://aws.amazon.com/blogs/aws/cross-region-read-replicas-for-amazon-rds-for-mysql/) or their applications have cross-region [disaster recovery](https://aws.amazon.com/disaster-recovery/) requirements.

Key management design with a multi-region database deployment must be carefully considered. Encrypted data under a single Region master key, that then propagates to a different Region, requires a cross-region AWS KMS call to decrypt the data key. This post encrypts the data key under a CMK in each Region

Each Region’s encrypted data key is stored with the encrypted data [in the AWS Encryption SDK encrypted message format](https://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/message-format.html), thus enabling you to avoid a cross-region AWS KMS call on decrypt. In addition, it is possible to avoid incurring a decryption dependency on a single Region because the data keys are encrypted for each Region.

Encrypting data under multi-region CMKs entails the following workflow. The application makes one KMS request to the local Region and then a subsequent AWS KMS request to each additional Region. Upon decryption, the application calls only to the local Region for decrypting the data key.

## AWS Encryption SDK message format reference

The encryption operations in the AWS Encryption SDK return a single data structure or [**encrypted message**](https://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/concepts.html#message) that contains the encrypted data (ciphertext) and all encrypted data keys. To understand this data structure, or to build libraries that read and write it, you need to understand the message format.

The message format consists of at least two parts: a header and a body. In some cases, the message format consists of a third part, a footer. The message format defines an ordered sequence of bytes in network byte order, also called big-endian format. The message format begins with the header, followed by the body, followed by the footer (when there is one).

The [**algorithms suites**](https://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/algorithms-reference.html) supported by the AWS Encryption SDK use one of two message format versions. Algorithm suites without [**key commitment**](https://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/concepts.html#key-commitment) use message format version 1. Algorithm suites with key commitment use message format version 2.

# Automation for this & EVRY topic.

## 5.1 Best practices for automating DB instance creation

It’s an Amazon RDS best practice to create a DB instance with the preferred minor version of the database engine. You can use the AWS CLI, Amazon RDS API, or AWS CloudFormation to automate DB instance creation. When you use these methods, you can specify only the major version and Amazon RDS automatically creates the instance with the preferred minor version. For example, if PostgreSQL 12.5 is the preferred minor version, and if you specify version 12 with create-db-instance, the DB instance will be version 12.5.

To determine the preferred minor version, you can run the describe-db-engine-versions command with the --default-only option as shown in the following example.

aws rds describe-db-engine-versions --default-only --engine postgres

{

"DBEngineVersions": [

{

"Engine": "postgres",

"EngineVersion": "12.5",

"DBParameterGroupFamily": "postgres12",

"DBEngineDescription": "PostgreSQL",

"DBEngineVersionDescription": "PostgreSQL 12.5-R1",

...some output truncated...

}

]

}

For information on creating DB instances programmatically, see the following resources:

## 5.2 Using the AWS CLI – [create-db-instance](https://docs.aws.amazon.com/cli/latest/reference/rds/create-db-instance.html)

To create a DB instance by using the AWS CLI, call the [create-db-instance](https://docs.aws.amazon.com/cli/latest/reference/rds/create-db-instance.html) command with the following parameters:

* --db-instance-identifier
* --db-instance-class
* --vpc-security-group-ids
* --db-subnet-group
* --engine
* --master-username
* --master-user-password
* --allocated-storage
* --backup-retention-period

For information about each setting, see [Settings for DB instances](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateDBInstance.html#USER_CreateDBInstance.Settings).

This example uses Microsoft SQL Server.

### 5.2.1 Example

For Linux, macOS, or Unix:

aws rds create-db-instance \

--engine *sqlserver-se* \

--db-instance-identifier *mymsftsqlserver* \

--allocated-storage *250* \

--db-instance-class *db.t3.large* \

--vpc-security-group-ids *mysecuritygroup* \

--db-subnet-group *mydbsubnetgroup* \

--master-username *masterawsuser* \

--master-user-password *masteruserpassword* \

--backup-retention-period *3*

For Windows:

aws rds create-db-instance ^

--engine *sqlserver-se* ^

--db-instance-identifier *mydbinstance* ^

--allocated-storage *250* ^

--db-instance-class *db.t3.large* ^

--vpc-security-group-ids *mysecuritygroup* ^

--db-subnet-group *mydbsubnetgroup* ^

--master-username *masterawsuser* ^

--master-user-password *masteruserpassword* ^

--backup-retention-period *3*

This command produces output similar to the following.

DBINSTANCE mydbinstance db.t3.large sqlserver-se 250 sa creating 3 \*\*\*\* n 10.50.2789

SECGROUP default active

PARAMGRP default.sqlserver-se-14 in-sync

* Using the Amazon RDS API – [CreateDBInstance](https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_CreateDBInstance.html)

## 5.3 Using AWS CloudFormation – [AWS::RDS::DBInstance](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-rds-database-instance.html)

When you use AWS CloudFormation, you work with templates and stacks. You create templates to describe your AWS resources and their properties. Whenever you create a stack, CloudFormation provisions the resources that are described in your template.

### 5.3.1Templates

A CloudFormation template is a JSON or YAML formatted text file. You can save these files with any extension, such as .json, .yaml, .template, or .txt. CloudFormation uses these templates as blueprints for building your AWS resources. For example, in a template, you can describe an Amazon EC2 instance, such as the instance type, the AMI ID, block device mappings, and its Amazon EC2 key pair name. Whenever you create a stack, you also specify a template that CloudFormation uses to create whatever you described in the template.

For example, if you created a stack with the following template, CloudFormation provisions an instance with an ami-0ff8a91507f77f867 AMI ID, t2.micro instance type, testkey key pair name, and an Amazon EBS volume.

**JSON**

{

"AWSTemplateFormatVersion": "2010-09-09",

"Description": "A sample template",

"Resources": {

"MyEC2Instance": {

"Type": "AWS::EC2::Instance",

"Properties": {

"ImageId": "ami-0ff8a91507f77f867",

"InstanceType": "t2.micro",

"KeyName": "testkey",

"BlockDeviceMappings": [

{

"DeviceName": "/dev/sdm",

"Ebs": {

"VolumeType": "io1",

"Iops": 200,

"DeleteOnTermination": false,

"VolumeSize": 20

}

}

]

}

}

}

}

# Encryption in transit

Previously we've seen how to encrypt an Aurora cluster at rest. Now we want to see how we can enforce encryption between the DB cluster and clients and ensure that all data transferred is encrypted.

## 6.1 [Enforce encryption in transit with RDS Aurora PostgreSQL](https://catalog.us-east-1.prod.workshops.aws/workshops/aad9ff1e-b607-45bc-893f-121ea5224f24/en-US/rds/aurora/3-encryption-in-transit" \l "enforce-encryption-in-transit-with-rds-aurora-postgresql)

Amazon RDS supports Secure Socket Layer (SSL - AES-256) encryption for Aurora MySQL/PostgreSQL DB clusters. Using SSL, you can encrypt a connection between your applications and your Aurora DB clusters.

### [Create custom Aurora PostgreSQL parameter group](https://catalog.us-east-1.prod.workshops.aws/workshops/aad9ff1e-b607-45bc-893f-121ea5224f24/en-US/rds/aurora/3-encryption-in-transit" \l "create-custom-aurora-postgresql-parameter-group)

1. On Services, click on **RDS**.
2. In the service menu click **Parameter groups**.
3. Click on **Create parameter group**.
4. Select **aurora-postgresql11** as group family.
5. As Type select **DB Cluster Parameter Group**.
6. As Group name type aurora-pg-ssl.
7. For description type SSL connection enforced.
8. Click on **Create**.
9. Click on the parameter group name aurora-pg-ssl.
10. On the parameter group configuration page, filter for rds.force\_ssl.
11. Select the parameter **rds.force\_ssl** and click **Edit parameters**.
12. Change the Value to **1** to enable SSL enforcement.
13. Click on **Save changes**.

## [6.2 Change the parameter group of the Aurora PostgreSQL database cluster](https://catalog.us-east-1.prod.workshops.aws/workshops/aad9ff1e-b607-45bc-893f-121ea5224f24/en-US/rds/aurora/3-encryption-in-transit" \l "change-the-parameter-group-of-the-aurora-postgresql-database-cluster)

1. In the service menu click **Databases**.
2. Click on the database name **database-1**.
3. Click on **Modify**.
4. In Additional configuration / DB cluster parameter group choose the custom parameter group **aurora-pg-ssl**.
5. Click on **continue**.
6. Select **Apply immediately**.
7. Click on **Modify cluster**.

For more information about RDS Aurora PostgreSQL using SSL see [Securing Aurora PostgreSQL data with SSL](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraPostgreSQL.Security.html#AuroraPostgreSQL.Security.SSL)

## [6.3 Connect to your encrypted cluster using Cloud9](https://catalog.us-east-1.prod.workshops.aws/workshops/aad9ff1e-b607-45bc-893f-121ea5224f24/en-US/rds/aurora/3-encryption-in-transit" \l "step-3:-connect-to-your-encrypted-cluster-using-cloud9)

1. On Services, click on **Cloud9**.
2. Click on **Create environment**.
3. For Name type workshop and click on **Next step**.
4. In the Environment settings do not modify anything and click on **Next step**.
5. Click on **Create environment**.
6. Go back to **Cloud9**, select your Cloud9 instance and click on **View details**.
7. Identity the Security groups associated to your Cloud9 instance.
8. On Services, click on **RDS**.
9. Click on **Databases**.
10. Click on one node of your RDS cluster.
11. In the Connectivity & security tab click on the security group associated to the cluster.
12. Click on **Actions/Edit inbound rules**.
13. Add the Cloud9 security group to the first line, allowing all traffic coming from the Cloud9 instance.
14. Click on **Save rules**.
15. On Services, click on **Cloud9**.
16. Select you Cloud9 instance and click on **Open IDE**.
17. Once the environment is ready, install the postgreSQL client:

sudo yum -u install postgresql

1. Connect to your RDS cluster (replace CLUSTER\_NODE with the endpoint name of one of your nodes):

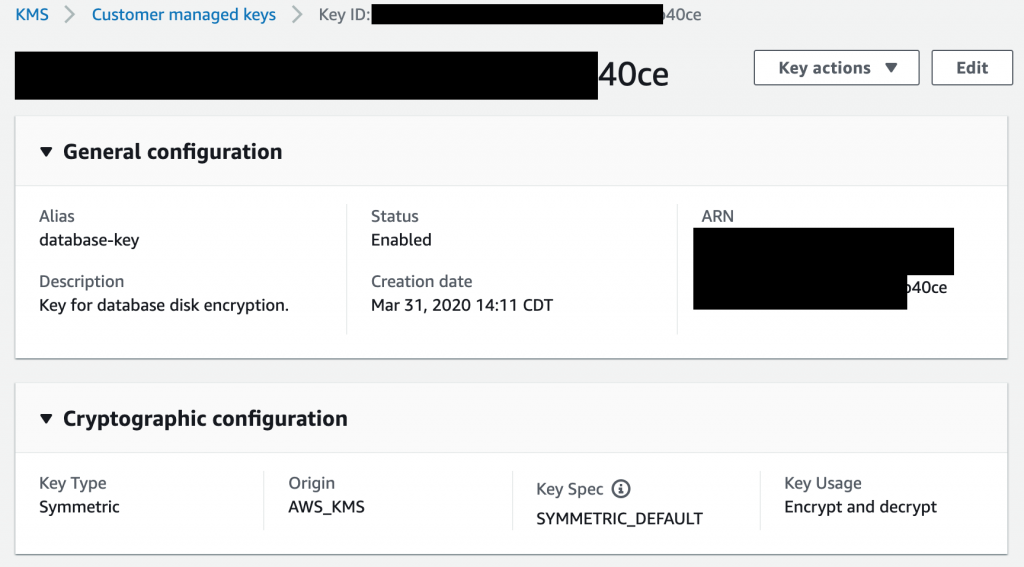
psql -h CLUSTER\_NODE -d workshop -U postgres

# Encryption at Rest – RDS

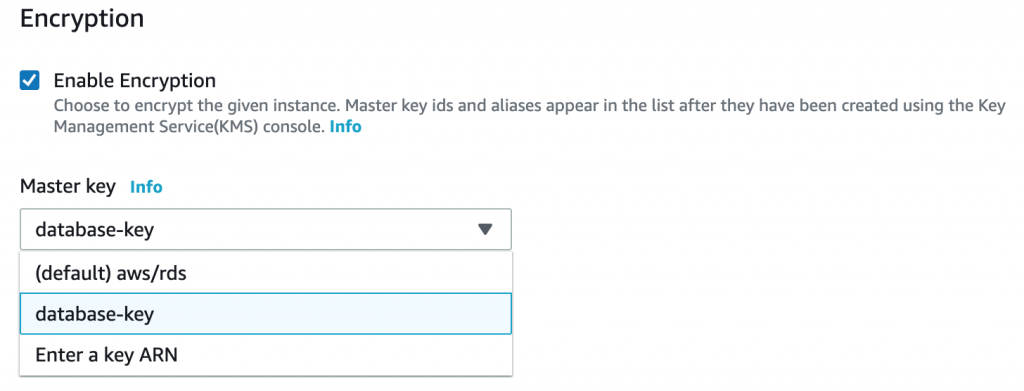
As RDS is a managed service, you do not interact with the base operating system or filesystem.  Rather, you leverage the API provided by AWS to launch and configure your instance. This poses a problem if you are currently using a **keyring\_\*** plugin in MySQL – how do you store your master key on the filesystem or configure other plugins for remote key management?  Unfortunately, you can’t. However, there is still a viable solution for encrypting your data at rest with RDS.

With RDS being built on the underlying IaaS components of AWS, you are able to leverage the existing functionality of those components through the API.  The feature we are looking at here is EBS (Elastic Block Store) encryption. Much like the full disk encryption method described above, EBS allows you to encrypt volumes using AWS default keys or customer master keys (CMKs) defined within the Key Management Service (KMS).  These keys are required for the volume to be usable and for snapshots to be restorable. This enables you to ensure data-at-rest encryption for your RDS instances.

In the past, I’ve used both the RDS default encryption key and user-defined keys.  For this exercise, however, I wanted to ensure I could prevent my database from being restored or restarted on demand.   To start, I created a new CMK in the KMS service to use with my database and gave the RDS service permission to access it:

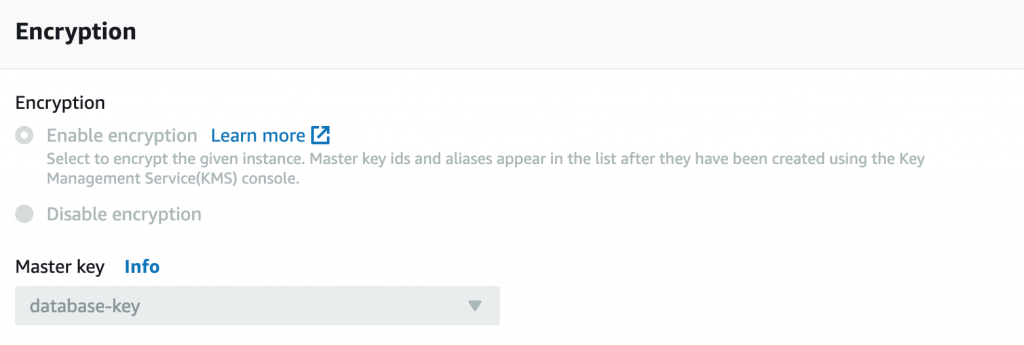


Then, when I create my RDS instance, I can choose this new key when I enable encryption.  For my test, I encrypted my instance using a cleverly named CMK key called database-key:

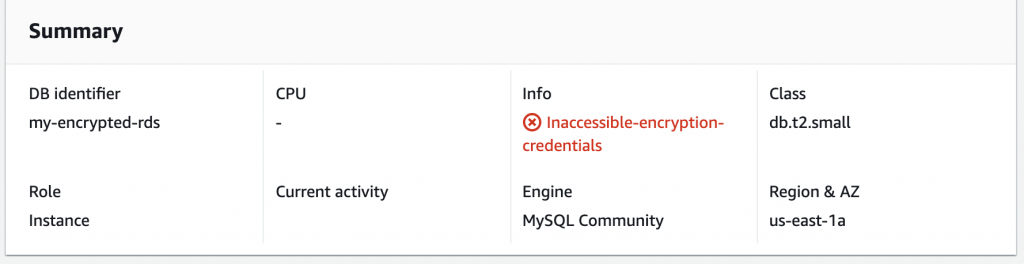


Note that along with my CMK, the (default) aws/rds key is an option.  I want control over my key and when it is used so I choose my key and not the default.  Note that certain instance types don’t offer encryption so if you don’t see this option when creating your instance, [check your instance type against the list here](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.Encryption.html#Overview.Encryption.Availability).

After that, it is just business as usual.  I connected to my RDS instance, populated some dummy data, and took my first snapshot.  This is when it was time to confirm that I could indeed disable the key and prevent access to my data.  Here is the important thing to note – as my instance was encrypted with a specific key at startup, the snapshots also require the same key.  You can see that in my Snapshot Restore window, it specifies the key needed and I don’t have the option to disable it or choose another key:

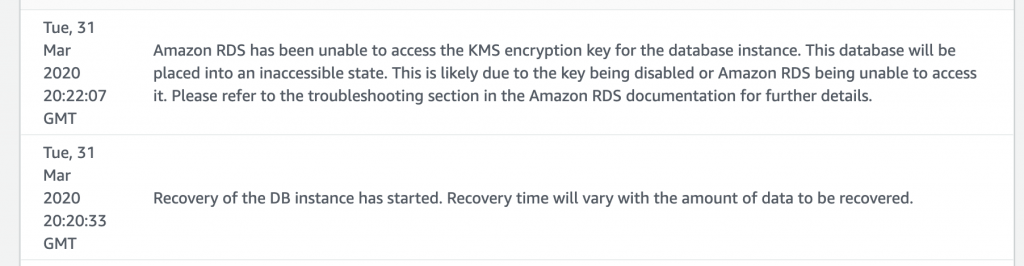


This is how RDS ensures that the key is tied to the instance and you can’t just take the data and restore it elsewhere.  After disabling my CMK (with my instance still running), I tried to restore my snapshot. Success! My snapshot **was not able** to be restored and I was pleasantly greeted with the following error message:



This confirms that by using my own key, I can effectively lock out any unauthorized restoration or access to the data at the disk level.  But now, I wanted to verify one additional item. Looking at how encryption at rest works on a standard instance, I also wanted to confirm that the key was needed to start the instance.

With the key disabled, I stopped the running instance.  After that process completed, I tried to restart my instance.  Again, it was successful in that I was **not able to restart my instance** with the key disabled:



So just like a standard instance that is missing the master key, the RDS instance isn’t able to start with the key disabled.  Similar to an on-prem deployment, ensuring that your CMK is secure is outside of the scope of this blog post. But with that being said, this exercise demonstrates the power and flexibility of using CMKs to ensure your data is encrypted at rest.

# To determine if your RDS database instances are encrypted

Perform the following:

## 8.1 Using AWS Console:

01Login to the AWS Management Console.

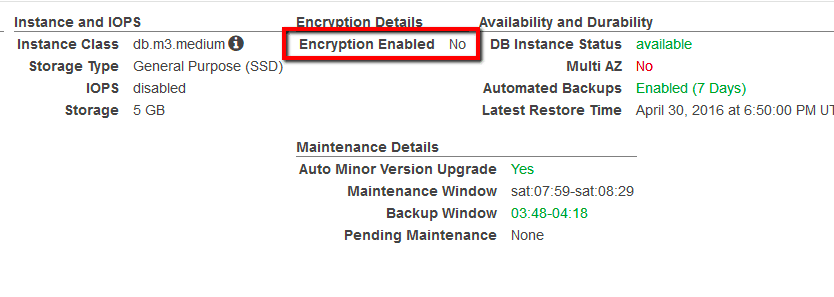
02Navigate to RDS dashboard at <https://console.aws.amazon.com/rds/>.

03In the navigation panel, under **RDS Dashboard**, click **Instances**.

04Select the RDS instance that you want to examine.

05Click **Instance Action**s button from the dashboard top menu and select **See Details**.

06Under **Encryption Details** section, search for the **Encryption Enabled** status:



If the current status is set to **No**, data-at-rest encryption is not enabled for the selected RDS database instance.

07Repeat steps no. 4 – 6 for each RDS instance provisioned in the current region. Change the AWS region from the navigation bar to repeat the process for other regions.

## 8.2Using AWS CLI

01Run **describe-db-instances** command (OSX/Linux/UNIX) to list all RDS database names, available in the selected AWS region (US East region in this case):

*aws rds describe-db-instances*

*--region* ***us-east-1***

*--query 'DBInstances[\*].DBInstanceIdentifier'*

02The command output should return each database instance identifier (name):

*[*

***"prod-mysql-db"***

*]*

03Run again **describe-db-instances** command (OSX/Linux/UNIX) using the RDS instance identifier returned earlier, to determine if the selected database instance is currently encrypted:

*aws rds describe-db-instances*

*--region* ***us-east-1***

*--db-instance-identifier* ***prod-mysql-db***

*--query 'DBInstances[\*].StorageEncrypted'*

04The command output should return the encryption status (as the **StorageEncrypted** parameter value) for the selected instance (true for enabled, false for disabled):

*aws rds describe-db-instances*

*--region [*

***false***

*]*

05If the **StorageEncrypted** parameter value is set to false, data-at-rest encryption is not enabled for the selected RDS database instance.

*aws rds describe-db-instances*

*--region [*

***false***

*]*

06Repeat steps no. 1 – 4 for each RDS instance provisioned in the current region. Change the AWS region by using the --region filter to repeat the process for other regions.

# To enable data encryption for your existing RDS instances

you need to re-create (back up and restore) them with encryption flag enabled, by performing the following steps:

## 9.1 Using AWS Console

01Login to the AWS Management Console.

02Navigate to RDS dashboard at <https://console.aws.amazon.com/rds/>.

03In the navigation panel, under **RDS Dashboard**, click **Instances**.

04Select the RDS database instance that you want to encrypt.

05Click **Instance Actions** button from the dashboard top menu and select **Take Snapshot**.

06On the **Take DB Snapshot** page, enter a name for the instance snapshot in the **Snapshot Name** field and click **Take Snapshot** (the backup process may take few minutes and depends on your instance storage size).

07Select the new created snapshot and click the **Copy Snapshot** button from the dashboard top menu.

08On the **Make Copy of DB Snapshot** page, perform the following:

1. In the **New DB Snapshot Identifier** field, enter a name for the new snapshot (copy).
2. Check **Copy Tags** so the new snapshot can have the same tags as the source snapshot.
3. Select **Yes** from the **Enable Encryption** dropdown list to enable encryption. You can choose to use the AWS default encryption key or your custom key (key ARN required) by selecting it from the **Master Key** dropdown list.

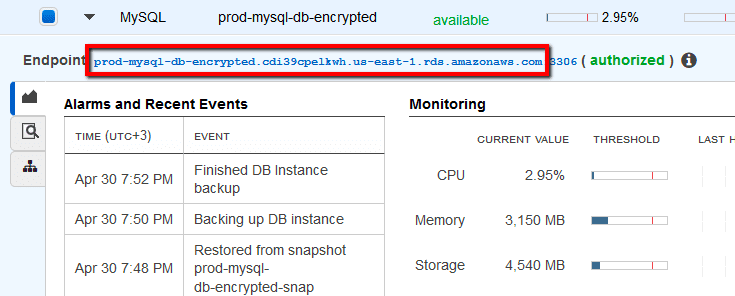
09Click **Copy Snapshot** to create an encrypted copy of the selected instance snapshot.

10Select the new snapshot copy (encrypted) and click **Restore Snapshot** button from the dashboard top menu. This will restore the encrypted snapshot to a new database instance.

11On the **Restore DB Instance** page, enter a unique name for the new database instance in the **DB Instance Identifier\*** field.

12Review the instance configuration details and click **Restore DB Instance**.

13As soon as the new instance provisioning process is completed (its status becomes **available**), you can update your application configuration to refer to the endpoint:



of the new (encrypted) database instance. Once the database endpoint is changed at your application level, you can remove the unecrypted instance.

14Repeat steps no. 4 – 13 for each RDS instance that you want to encrypt, available in the current region. Change the AWS region from the navigation bar to repeat the process for other regions.

## 9.2 Using AWS CLI

01Run **describe-db-instances** command (OSX/Linux/UNIX) to list all RDS database names (identifiers), available in the selected AWS region:

*aws rds describe-db-instances*

*--region* ***us-east-1***

*--query 'DBInstances[\*].DBInstanceIdentifier'*

02The command output should return each database instance identifier:

*[*

***"prod-mysql-db"***

*]*

03Run **create-db-snapshot** command (OSX/Linux/UNIX) to create a snapshot for the selected database instance. The following command example creates a snapshot named prod-mysql-db-snapshot from an RDS instance named prod-mysql-db:

*aws rds create-db-snapshot*

*--region us-east-1*

*--db-snapshot-identifier prod-mysql-db-snapshot*

*--db-instance-identifier* ***prod-mysql-db***

04The command output should return the new snapshot metadata:

*{*

*"DBSnapshot": {*

*"Engine": "mysql",*

*"Status": "creating",*

*"AvailabilityZone": "us-east-1b",*

*"PercentProgress": 0,*

*"MasterUsername": "webappdb",*

*"Encrypted": false,*

*"LicenseModel": "general-public-license",*

*"StorageType": "gp2",*

*"VpcId": "vpc-f7ac5792",*

*"DBSnapshotIdentifier": "prod-mysql-db-snapshot",*

*"InstanceCreateTime": "2016-04-30T15:44:26.042Z",*

*"OptionGroupName": "default:mysql-5-6",*

*"AllocatedStorage": 5,*

*"EngineVersion": "5.6.27",*

*"SnapshotType": "manual",*

*"Port": 3306,*

*"DBInstanceIdentifier": "prod-mysql-db"*

*}*

*}*

05Now run **list-aliases** command (OSX/Linux/UNIX) to list the KMS keys aliases (names) available in specified region:

*aws kms list-aliases*

*--region us-east-1*

06The command output should return each key alias currently available. For our RDS encryption activation process, locate the ID (highlighted) of the AWS default KMS key provided for database encryption (alias/aws/rds)

*{*

*"Aliases": [*

*{*

*"AliasArn": "arn:aws:kms:us-east-1:123456789012:alias/aws/ebs",*

*"AliasName": "alias/aws/ebs",*

*"TargetKeyId": "d6c03026-b0bd-451e-a864-a68355f4f035"*

*},*

*{*

*"AliasArn": "arn:aws:kms:us-east-1:123456789012:alias/aws/rds",*

*"AliasName": "alias/aws/rds",*

*"TargetKeyId":* ***"8d8d3ab9-db2a-428f-b82e-d38cb05ce1a4"***

*},*

*{*

*"AliasArn": "arn:aws:kms:us-east-1:123456789012:alias/aws/s3",*

*"AliasName": "alias/aws/s3"*

*}*

*]*

*}*

07Run **copy-db-snapshot** command (OSX/Linux/UNIX) using the default KMS key ID for RDS instances returned earlier to create an encrypted copy of the database instance snapshot:

*aws rds copy-db-snapshot*

*--region us-east-1*

*--source-db-snapshot-identifier prod-mysql-db-snapshot*

*--target-db-snapshot-identifier prod-mysql-db-snapshot-encrypted*

*--copy-tags*

*--kms-key-id* ***8d8d4bg8-db2a-4268f-b52e-3dbab05ce9a5***

08The command output should return the encrypted instance snapshot (copy) metadata:

*{*

*"DBSnapshot": {*

*"Engine": "mysql",*

*"Status": "creating",*

*"AvailabilityZone": "us-east-1b",*

*"SourceRegion": "us-east-1",*

*"PercentProgress": 0,*

*"MasterUsername": "webappdb",*

*"Encrypted": true,*

*"LicenseModel": "general-public-license",*

*"StorageType": "gp2",*

*"KmsKeyId": "arn:aws:kms:us-east-1:123456789012:*

*key/8d8d3ab9-db2a-428f-b82e-d38cb05ce1a4",*

*"VpcId": "vpc-f7ac5792",*

*"SourceDBSnapshotIdentifier": "arn:aws:rds:us-east-1:123456789012:*

*snapshot:prod-mysql-db-snapshot",*

*"DBSnapshotIdentifier": "prod-mysql-db-snapshot-encrypted",*

*"InstanceCreateTime": "2016-04-30T15:44:26.042Z",*

*"OptionGroupName": "default:mysql-5-6",*

*"AllocatedStorage": 5,*

*"EngineVersion": "5.6.27",*

*"SnapshotType": "manual",*

*"Port": 3306,*

*"DBInstanceIdentifier": "prod-mysql-db"*

*}*

*}*

09Run **restore-db-instance-from-db-snapshot** command (OSX/Linux/UNIX) to restore the encrypted snapshot created at the previous step to a new database instance:

*aws rds restore-db-instance-from-db-snapshot*

*--region us-east-1*

*--db-instance-identifier prod-mysql-db-encrypted*

*--db-snapshot-identifier prod-mysql-db-snapshot-encrypted*

10If successful, the command output should return the new encrypted database instance metadata:

*{*

*"DBInstance": {*

*"PubliclyAccessible": true,*

*"MasterUsername": "webappdb",*

*"MonitoringInterval": 0,*

*"LicenseModel": "general-public-license",*

*...*

*"DbiResourceId": "db-GPXVANNOGAXV5BMSEF2U2JEW5A",*

*"CACertificateIdentifier": "rds-ca-2015",*

*"KmsKeyId": "arn:aws:kms:us-east-1:123456789012:*

*key/8d8d3ab9-db2a-428f-b82e-d38cb05ce1a4",*

***"StorageEncrypted": true,***

*"DBInstanceClass": "db.m3.medium",*

*"DbInstancePort": 0,*

*"DBInstanceIdentifier": "prod-mysql-db-encrypted"*

*}*

*}*

11Run **describe-db-instances** command (OSX/Linux/UNIX) to make sure the new database instance is encrypted:

*aws rds describe-db-instances*

*--region* ***us-east-1***

*--db-instance-identifier* ***prod-mysql-db-encrypted***

*--query 'DBInstances[\*].StorageEncrypted'*

12The command output should return the encryption status (as the **StorageEncrypted** parameter value) for the selected instance (true for enabled, false for disabled):

*[*

***true***

*]*

13Repeat steps no. 1 – 12 for each RDS instance provisioned in the current region. Change the AWS region by using the --region filter to repeat the process for other regions.

**Implementation**

C:\Users\User>aws rds create-db-instance --engine MySQL --db-instance-identifier database-2 --allocated-storage 200 --db-instance-class db.t3.micro --vpc-security-group-ids sg-0c9c7803eb60d609d --db-subnet-group default-vpc-0ad7e0bf54c1ba342 --master-username admin --master-user-password hemapaul --backup-retention-period 3

DBINSTANCE 200 True 3 region rds-ca-2019 False False arn:aws:rds:us-east-1:600148095221:db:database-2 db.t3.micro database-2 creating 0 db-BQHXI6VZVOHV5JCGE2RVMVBCOU False mysql 8.0.28 False general-public-license admin 0 False False 07:28-07:58 mon:09:16-mon:09:46 False False gp2

DBPARAMETERGROUPS default.mysql8.0 in-sync

DBSUBNETGROUP Created from the RDS Management Console default-vpc-0ad7e0bf54c1ba342 Complete vpc-0ad7e0bf54c1ba342

SUBNETS subnet-003a9fb11da683267 Active

SUBNETAVAILABILITYZONE us-east-1b

SUBNETS subnet-0c336682aa104b7bd Active

SUBNETAVAILABILITYZONE us-east-1a

SUBNETS subnet-07e75c284c9d065b3 Active

SUBNETAVAILABILITYZONE us-east-1f

SUBNETS subnet-027fd59eca648be3b Active

SUBNETAVAILABILITYZONE us-east-1d

SUBNETS subnet-0267b62826279b2b7 Active

SUBNETAVAILABILITYZONE us-east-1e

SUBNETS subnet-00de8494e22669aec Active

SUBNETAVAILABILITYZONE us-east-1c

OPTIONGROUPMEMBERSHIPS default:mysql-8-0 in-sync

PENDINGMODIFIEDVALUES \*\*\*\*

VPCSECURITYGROUPS active sg-0c9c7803eb60d609d

C:\Users\User>aws rds describe-db-instances --db-instance-identifier database-2 --query "\*[].{StorageEncrypted:StorageEncrypted}" --output text

false

C:\Users\User>aws rds create-db-snapshot --region us-east-1 --db-snapshot-identifier database-2-db-snapshot --db-instance-identifier database-2

DBSNAPSHOT 200 us-east-1d database-2 arn:aws:rds:us-east-1:600148095221:snapshot:database-2-db-snapshot

database-2-db-snapshot db-BQHXI6VZVOHV5JCGE2RVMVBCOU False mysql 8.0.28 False 2022-10-28T06:58:10.986000+00:00

general-public-license admin default:mysql-8-0 0 3306 region manual creating gp2 vpc-0ad7e0bf54c1ba342

C:\Users\User>aws kms list-aliases --region us-east-1

ALIASES arn:aws:kms:us-east-1:600148095221:alias/aws/dynamodb alias/aws/dynamodb

ALIASES arn:aws:kms:us-east-1:600148095221:alias/aws/ebs alias/aws/ebs 2022-09-26T11:44:18.033000+05:30 2022-09-26T11:44:18.033000+05:30 2e09e237-fc6b-4712-bfa5-6f43a88bb784

ALIASES arn:aws:kms:us-east-1:600148095221:alias/aws/elasticfilesystem alias/aws/elasticfilesystem ALIASES arn:aws:kms:us-east-1:600148095221:alias/aws/es alias/aws/es

ALIASES arn:aws:kms:us-east-1:600148095221:alias/aws/glue alias/aws/glue

ALIASES arn:aws:kms:us-east-1:600148095221:alias/aws/kinesisvideo alias/aws/kinesisvideo

ALIASES arn:aws:kms:us-east-1:600148095221:alias/aws/rds alias/aws/rds 2022-10-28T10:05:11.810000+05:30 2022-10-28T10:05:11.810000+05:30 b337bfb7-acef-4d01-b0dd-09d68155358e

ALIASES arn:aws:kms:us-east-1:600148095221:alias/aws/redshift alias/aws/redshift

ALIASES arn:aws:kms:us-east-1:600148095221:alias/aws/redshifttest alias/aws/redshifttest

ALIASES arn:aws:kms:us-east-1:600148095221:alias/aws/s3 alias/aws/s3

ALIASES arn:aws:kms:us-east-1:600148095221:alias/aws/secretsmanager alias/aws/secretsmanager 2022-09-28T17:47:38.010000+05:30 2022-09-28T17:47:38.010000+05:30 66d232f4-824a-4944-a0e9-ac7f4f1f5dbf

ALIASES arn:aws:kms:us-east-1:600148095221:alias/aws/ssm alias/aws/ssm

ALIASES arn:aws:kms:us-east-1:600148095221:alias/aws/xray alias/aws/xray

ALIASES arn:aws:kms:us-east-1:600148095221:alias/kmsmacie alias/kmsmacie 2022-10-04T16:29:02.344000+05:30 2022-10-04T16:29:02.344000+05:30 29e3bb5e-de48-4bed-8d66-14a07597e8f9

C:\Users\User>aws rds copy-db-snapshot --region us-east-1 --source-db-snapshot-identifier database-2-db-snapshot --target-db-snapshot-identifier database-2-db-snapshot-encrypted --copy-tags --kms-key-id b337bfb7-acef-4d01-b0dd-09d68155358e

DBSNAPSHOT 200 us-east-1d database-2

arn:aws:rds:us-east-1:600148095221:snapshot:database-2-db-snapshot-encrypted

database-2-db-snapshot-encrypted

db-BQHXI6VZVOHV5JCGE2RVMVBCOU True

mysql 8.0.28 False 2022-10-28T06:58:10.986000+00:00

arn:aws:kms:us-east-1:600148095221:key/b337bfb7-acef-4d01-b0dd-09d68155358e

general-public-license admin default:mysql-8-0 2022-10-28T07:10:30.687000+00:00

0 3306 region manual arn:aws:rds:us-east-1:600148095221:snapshot:database-2-db-snapshot

us-east-1 creating gp2 vpc-0ad7e0bf54c1ba342

C:\Users\User>aws rds restore-db-instance-from-db-snapshot --region us-east-1 --db-instance-identifier database-2-db-encrypted --db-snapshot-identifier database-2-db-snapshot-encrypted

DBINSTANCE 200 True 3 region rds-ca-2019 False False

arn:aws:rds:us-east-1:600148095221:db:database-2-db-encrypted db.t3.micro database-2-db-encrypted creating 0

db-WKHEPBPR7ARRGK3DXRV5SQ2NSI False mysql 8.0.28 False arn:aws:kms:us-east-1:600148095221:key/b337bfb7-acef-4d01-b0dd-09d68155358e

general-public-license admin 0 False False 07:28-07:58 mon:09:16-mon:09:46 True True gp2

DBPARAMETERGROUPS default.mysql8.0 in-sync

DBSUBNETGROUP default default Complete vpc-0ad7e0bf54c1ba342

SUBNETS subnet-003a9fb11da683267 Active

SUBNETAVAILABILITYZONE us-east-1b

SUBNETS subnet-0c336682aa104b7bd Active

SUBNETAVAILABILITYZONE us-east-1a

SUBNETS subnet-07e75c284c9d065b3 Active

SUBNETAVAILABILITYZONE us-east-1f

SUBNETS subnet-027fd59eca648be3b Active

SUBNETAVAILABILITYZONE us-east-1d

SUBNETS subnet-0267b62826279b2b7 Active

SUBNETAVAILABILITYZONE us-east-1e

SUBNETS subnet-00de8494e22669aec Active

SUBNETAVAILABILITYZONE us-east-1c

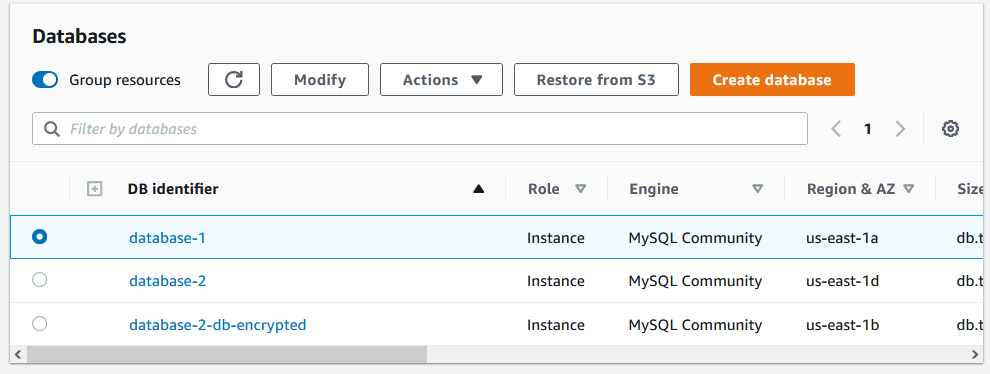
OPTIONGROUPMEMBERSHIPS default:mysql-8-0 pending-apply

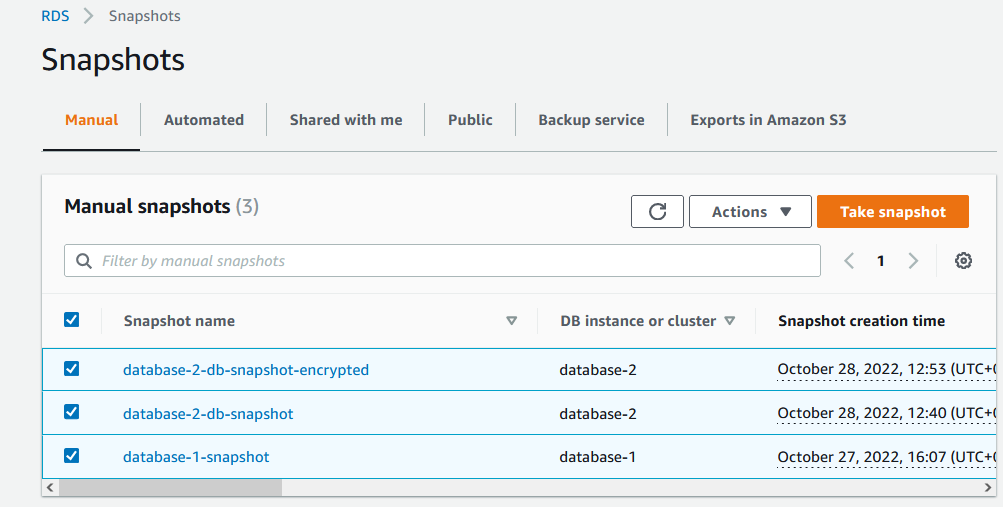
VPCSECURITYGROUPS active sg-0c9c7803eb60d609d

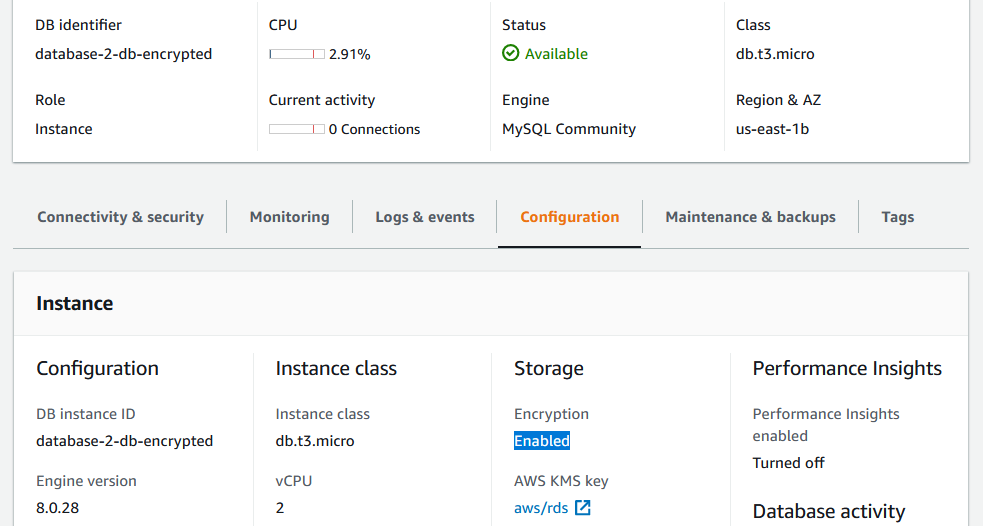
C:\Users\User>aws rds describe-db-instances --db-instance-identifier database-2-db-encrypted --query "\*[].{StorageEncrypted:StorageEncrypted}" --output text

True

Looking in the console, it is seen the instance is created and got encrypted.







# Limitations of Amazon RDS encrypted DB instances

* You can only encrypt an Amazon RDS DB instance when you create it, not after the DB instance is created.

However, because you can encrypt a copy of an unencrypted snapshot, you can effectively add encryption to an unencrypted DB instance. That is, you can create a snapshot of your DB instance, and then create an encrypted copy of that snapshot. You can then restore a DB instance from the encrypted snapshot, and thus you have an encrypted copy of your original DB instance. For more information, see [Copying a DB snapshot](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CopySnapshot.html).

* You can't turn off encryption on an encrypted DB instance.
* You can't create an encrypted snapshot of an unencrypted DB instance.
* A snapshot of an encrypted DB instance must be encrypted using the same KMS key as the DB instance.
* You can't have an encrypted read replica of an unencrypted DB instance or an unencrypted read replica of an encrypted DB instance.
* Encrypted read replicas must be encrypted with the same KMS key as the source DB instance when both are in the same AWS Region.
* You can't restore an unencrypted backup or snapshot to an encrypted DB instance.
* 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 RDS 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 instance. However, you can export data from an encrypted DB instance and import the data into an unencrypted DB instance.

# 11.Links:

<https://aws.amazon.com/premiumsupport/knowledge-center/update-encryption-key-rds/#:~:text=You%20can't%20change%20the,practices%20for%20working%20with%20PostgreSQL>.

<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_DeleteInstance.html>

<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_BestPractices.html#CHAP_BestPractices.PostgreSQL>

<https://docs.aws.amazon.com/cli/latest/reference/rds/create-db-instance.html>

<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateDBInstance.html>

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html>

<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.Encryption.html#Overview.Encryption.Limitations>

<https://aws.amazon.com/blogs/database/securing-data-in-amazon-rds-using-aws-kms-encryption/#:~:text=First%2C%20choose%20your%20destination%20Region,the%20RDS%20snapshot%20to%20copy>.

<https://www.trendmicro.com/cloudoneconformity/knowledge-base/aws/RDS/rds-encryption-enabled.html>

<https://www.percona.com/blog/2020/04/15/whats-the-best-way-to-enable-and-test-encryption-at-rest-in-rds/>

<https://catalog.us-east-1.prod.workshops.aws/workshops/aad9ff1e-b607-45bc-893f-121ea5224f24/en-US/rds/aurora/3-encryption-in-transit>

<https://aws.amazon.com/blogs/database/performing-sql-database-client-side-encryption-for-multi-region-high-availability/>

.