Amazon Redshift Test Cases

Last updated: 27 September 2018

Version: 0.1

**Document Control**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Document Title | | Version | Author | Summary |
| Customer – AWS Redshift Test Cases | v0.1 | |  | This document provides an introduction to the Amazon Redshift service and additional AWS services that interact with it. Test cases are provided with various scenarios on how to accomplish particular tasks with command line examples, AWS CloudTrail events, and AWS CloudFormation snippets when applicable. |

Table of Contents

[1. Executive Summary 4](#_Toc2273135)

[2. Test Cases 7](#_Toc2273136)

[2.1 Encryption-at-rest 7](#_Toc2273137)

[2.1.1 Redshift Data Should Be Encrypted 8](#_Toc2273138)

[2.1.2 KMS Keys Should Not Be Default Key 12](#_Toc2273139)

[2.1.3 KMS Keys Must Use Imported Key Material 18](#_Toc2273140)

[2.2 Encryption-in-transit 21](#_Toc2273141)

[2.2.1 SSL Should Be Enabled 21](#_Toc2273142)

[2.2.2 AWS Certificate Manager Should Have Imported customer Key Material 24](#_Toc2273143)

[2.2.3 Parameter “use\_fips\_ssl” Set To “true” 27](#_Toc2273144)

[2.2.4 Parameter “require\_SSL” Set To “true” 31](#_Toc2273145)

[2.2.5 Certificate Chain of Trust Should Only Include Whitelisted Authorities 32](#_Toc2273146)

[2.3 Encryption Key Management 33](#_Toc2273147)

[2.3.1 Key Rotation Policy 33](#_Toc2273148)

[2.4 Infrastructure 36](#_Toc2273149)

[2.4.1 VPC Flow Logs Enabled on Leadernode ENI 36](#_Toc2273150)

[2.4.2 Leader node Security Group Does Not Contain Outbound Rules 38](#_Toc2273151)

[2.4.3 No EIP Should Be Assigned to Redshift Cluster 41](#_Toc2273152)

[2.4.4 VPC Endpoint Enabled for S3 Access 43](#_Toc2273153)

[2.5 Identity and Access Management (IAM) 45](#_Toc2273154)

[2.5.1 Only Master and Superuser Local Users Exist 45](#_Toc2273155)

[2.5.2 Storing and Retrieving Passwords To Parameter Store With KMS 46](#_Toc2273156)

[2.5.3 IAM Role Policies Should Include “redshift:RequestTag” Condition 48](#_Toc2273157)

[2.5.4 Redshift Clusters Must Use Service Linked Role 50](#_Toc2273158)

[2.5.5 Database Users Should Have Password Disabled 54](#_Toc2273159)

[2.5.6 Snapshots Copied To Another Region Are Re-tagged With New Region 55](#_Toc2273160)

[Logging and Monitoring 58](#_Toc2273161)

[2.6.1 Confirm “enable\_user\_activity\_logging” Is Enabled and Logs Are Delivered To S3 58](#_Toc2273162)

[2.6.2 Redshift Cluster And Databases Should Not Use Default Parameter Groups 61](#_Toc2273163)

# Executive Summary

This document is an overview of the Amazon Redshift Service with test cases and scenarios to assist in security baselining. Each test case is provided with a scenario(s) that satisfies the request and could additionally provide a situation that a user should be aware of for their security posture. Scenarios also provide relevant AWS Command Line Interface (AWS CLI) commands, outputs, AWS CloudFormation snippets, and AWS CloudTrail events when available.

Amazon Redshift is a fast, scalable data warehouse that makes it simple and cost-effective to analyze all your data across your data warehouse and data lake. Redshift delivers ten times faster performance than other data warehouses by using machine learning, massively parallel query execution, and columnar storage on high-performance disk. You can setup and deploy a new data warehouse in minutes, and run queries across petabytes of data in your Redshift data warehouse, and exabytes of data in your data lake built on Amazon S3.

The Amazon Redshift service manages all of the work of setting up, operating, and scaling a data warehouse. These tasks include provisioning capacity, monitoring and backing up the cluster, and applying patches and upgrades to the Amazon Redshift engine. An Amazon Redshift cluster is a set of nodes, which consists of a leader node and one or more compute nodes. The type and number of compute nodes that you need depends on the size of your data, the number of queries you will execute, and the query execution performance that you need.

Depending on your data warehousing needs, you can start with a small, single-node cluster and easily scale up to a larger, multi-node cluster as your requirements change. You can add or remove compute nodes to the cluster with minimal interruption to the service. If you intend to keep your cluster running for a year or longer, you can save money by reserving compute nodes for a one-year or three-year period. Reserving compute nodes offers significant savings compared to the hourly rates that you pay when you provision compute nodes on demand. Snapshots are point-in-time backups of a cluster. There are two types of snapshots: automated and manual. Amazon Redshift stores these snapshots internally in Amazon Simple Storage Service (Amazon S3) by using an encrypted Secure Sockets Layer (SSL) connection. If you need to restore from a snapshot, Amazon Redshift creates a new cluster and imports data from the snapshot that you specify.

There are several features related to cluster access and security in Amazon Redshift. These features help you to control access to your cluster, define connectivity rules, and encrypt data and connections. These features are in addition to features related to database access and security in Amazon Redshift.

By default, an Amazon Redshift cluster is only accessible to the AWS account that creates the cluster. The cluster is locked down so that no one else has access. Within your AWS account, you use the AWS Identity and Access Management (IAM) service to create user accounts and manage permissions for those accounts to control cluster operations.

By default, any cluster that you create is closed to everyone. IAM credentials only control access to the Amazon Redshift API-related resources: The Amazon Redshift console, command line interface (CLI), API, and SDK. To enable access to the cluster from SQL client tools via JDBC or ODBC, you use security groups. If you are using the EC2-Classic platform for your Amazon Redshift cluster, you must use Amazon Redshift security groups. If you are using the EC2-VPC platform for your Amazon Redshift cluster, you must use VPC security groups. In either case, you add rules to the security group to grant explicit inbound access to a specific range of CIDR/IP addresses or to an Amazon Elastic Compute Cloud (Amazon EC2) security group if your SQL client runs on an Amazon EC2 instance. In addition to the inbound access rules, you create database users to provide credentials to authenticate to the database within the cluster itself.

When you provision the cluster, you can optionally choose to encrypt the cluster for additional security. When you enable encryption, Amazon Redshift stores all data in user-created tables in an encrypted format. You can use AWS Key Management Service (AWS KMS) to manage your Amazon Redshift encryption keys. Encryption is an immutable property of the cluster. The only way to switch from an encrypted cluster to a nonencrypted cluster is to unload the data and reload it into a new cluster. Encryption applies to the cluster and any backups. When you restore a cluster from an encrypted snapshot, the new cluster is encrypted as well. You can use Secure Sockets Layer (SSL) encryption to encrypt the connection between your SQL client and your cluster.

There are several features related to monitoring in Amazon Redshift. You can use database audit logging to generate activity logs, configure events and notification subscriptions to track information of interest, and use the metrics in Amazon Redshift and Amazon CloudWatch to learn about the health and performance of your clusters and databases.

You can use the database audit logging feature to track information about authentication attempts, connections, disconnections, changes to database user definitions, and queries run in the database. This information is useful for security and troubleshooting purposes in Amazon Redshift. The logs are stored in Amazon S3 buckets.

Amazon Redshift tracks events and retains information about them for a period of several weeks in your AWS account. For each event, Amazon Redshift reports information such as the date the event occurred, a description, the event source (for example, a cluster, a parameter group, or a snapshot), and the source ID. You can create Amazon Redshift event notification subscriptions that specify a set of event filters. When an event occurs that matches the filter criteria, Amazon Redshift uses Amazon Simple Notification Service to actively inform you that the event has occurred.

Amazon Redshift provides performance metrics and data so that you can track the health and performance of your clusters and databases. Amazon Redshift uses Amazon CloudWatch metrics to monitor the physical aspects of the cluster, such as CPU utilization, latency, and throughput. Amazon Redshift also provides query and load performance data to help you monitor the database activity in your cluster.

Amazon Redshift creates one database when you provision a cluster. This is the database you use to load data and run queries on your data. You can create additional databases as needed by running SQL commands. When you provision a cluster, you specify a master user who has access to all of the databases that are created within the cluster. This master user is a superuser who is the only user with access to the database initially, though this user can create additional superusers and users. Amazon Redshift uses parameter groups to define the behavior of all databases in a cluster, such as date presentation style and floating-point precision. If you don’t specify a parameter group when you provision your cluster, Amazon Redshift associates a default parameter group with the cluster.

Amazon Redshift is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon Redshift. CloudTrail captures all API calls for Amazon Redshift as events, including calls from the Amazon Redshift console and from code calls to the Amazon Redshift APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon Redshift. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in Event history. Using the information collected by CloudTrail, you can determine the request that was made to Amazon Redshift, the IP address from which the request was made, who made the request, when it was made, and additional details. All Amazon Redshift actions are logged by CloudTrail and are documented in the [Amazon Redshift API Reference](https://docs.aws.amazon.com/redshift/latest/APIReference/).

# Test Cases

Redshift Test Cases to assist with security baselining

## Encryption-at-rest

In Amazon Redshift, you can enable database encryption for your clusters to help protect data at rest. When you enable encryption for a cluster, the data blocks and system metadata are encrypted for the cluster and its snapshots. Encryption is an optional, immutable setting of a cluster. If you want encryption, you enable it during the cluster launch process. To go from an unencrypted cluster to an encrypted cluster or the other way around, unload your data from the existing cluster and reload it in a new cluster with the chosen encryption setting. For more information, see [Migrating an Unencrypted Cluster to an Encrypted Cluster](https://docs.aws.amazon.com/redshift/latest/mgmt/migrating-to-an-encrypted-cluster.html)

Though encryption is an optional setting in Amazon Redshift, we recommend enabling it for clusters that contain sensitive data. Additionally, you might be required to use encryption depending on the guidelines or regulations that govern your data. For example, the Payment Card Industry Data Security Standard (PCI DSS), the Sarbanes-Oxley Act (SOX), the Health Insurance Portability and Accountability Act (HIPAA), and other such regulations provide guidelines for handling specific types of data.

When you choose AWS KMS for key management with Amazon Redshift, there is a four-tier hierarchy of encryption keys. These keys, in hierarchical order, are the master key, a cluster encryption key (CEK), a database encryption key (DEK), and data encryption keys.

When you launch your cluster, Amazon Redshift returns a list of the customer master keys (CMKs) that your AWS account has created or has permission to use in AWS KMS. You select a CMK to use as your master key in the encryption hierarchy. By default, Amazon Redshift selects your default key as the master key. Your default key is an AWS-managed key that is created for your AWS account to use in Amazon Redshift. AWS KMS creates this key the first time you launch an encrypted cluster in a region and choose the default key.

For some customers policy a default key should not be used on Critical or Highly Sensitive data, you must have (or create) a customer-managed CMK separately in AWS KMS before you launch your cluster in Amazon Redshift. Customer-managed CMKs give you more flexibility, including the ability to create, rotate, disable, define access control for, and audit the encryption keys used to help protect your data. For more information about creating CMKs, go to [Creating Keys](https://docs.aws.amazon.com/kms/latest/developerguide/create-keys.html) in the *AWS Key Management Service Developer Guide*.

After you choose a master key, Amazon Redshift requests that AWS KMS generate a data key and encrypt it using the selected master key. This data key is used as the CEK in Amazon Redshift. AWS KMS exports the encrypted CEK to Amazon Redshift, where it is stored internally on disk in a separate network from the cluster along with the grant to the CMK and the encryption context for the CEK. Only the encrypted CEK is exported to Amazon Redshift; the CMK remains in AWS KMS. Amazon Redshift also passes the encrypted CEK over a secure channel to the cluster and loads it into memory. Then, Amazon Redshift calls AWS KMS to decrypt the CEK and loads the decrypted CEK into memory. Next, Amazon Redshift randomly generates a key to use as the DEK and loads it into memory in the cluster. The decrypted CEK is used to encrypt the DEK, which is then passed over a secure channel from the cluster to be stored internally by Amazon Redshift on disk in a separate network from the cluster. Like the CEK, both the encrypted and decrypted versions of the DEK are loaded into memory in the cluster. The decrypted version of the DEK is then used to encrypt the individual encryption keys that are randomly generated for each data block in the database.

When the cluster reboots, Amazon Redshift starts with the internally stored, encrypted versions of the CEK and DEK, reloads them into memory, and then calls AWS KMS to decrypt the CEK with the CMK again so it can be loaded into memory. The decrypted CEK is then used to decrypt the DEK again, and the decrypted DEK is loaded into memory and used to encrypt and decrypt the data block keys as needed.

In Amazon Redshift, you can rotate encryption keys for encrypted clusters. When you start the key rotation process, Amazon Redshift rotates the CEK for the specified cluster and for any automated or manual snapshots of the cluster. Amazon Redshift also rotates the DEK for the specified cluster, but cannot rotate the DEK for the snapshots while they are stored internally in Amazon Simple Storage Service (Amazon S3) and encrypted using the existing DEK.

While the rotation is in progress, the cluster is put into a ROTATING\_KEYS state until completion, at which time the cluster returns to the AVAILABLE state. Amazon Redshift handles decryption and re-encryption during the key rotation process. Before you delete a cluster, consider whether its snapshots rely on key rotation as you cannot rotate keys for snapshots without a source cluster. Because the cluster is momentarily unavailable during the key rotation process, you should rotate keys only as often as your data needs require or when you suspect the keys might have been compromised. As a best practice, you should review the type of data that you store and plan how often to rotate the keys that encrypt that data. The frequency for rotating keys varies depending on your corporate policies for data security, and any industry standards regarding sensitive data and regulatory compliance. Ensure that your plan balances security needs with availability considerations for your cluster.

### Redshift Data Should Be Encrypted

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Load data in to Amazon Redshift which should be encrypted | I can verify that data is encrypted at rest |

Encryption is an optional, immutable setting of a cluster. If you want encryption, you enable it during the cluster launch process. To go from an unencrypted cluster to an encrypted cluster or the other way around, unload your data from the existing cluster and reload it in a new cluster with the chosen encryption setting. For more information, see [Migrating an Unencrypted Cluster to an Encrypted Cluster](https://docs.aws.amazon.com/redshift/latest/mgmt/migrating-to-an-encrypted-cluster.html)

#### Scenario – Launching an Amazon Redshift cluster with encryption enabled

##### CLI Command

When using the AWS CLI, you can encrypt the Redshift cluster using the “--encrypted” option. This will enable KMS encryption on the Redshift cluster using a default KMS Key. Some customer policy, default KMS keys should not be utilized. In that case you can add the “--kms-key-id” option and provide a kms key id that can be used.

aws redshift create-cluster --node-type ds2.xlarge --number-of-nodes 2 --master-username adminuser --master-user-pass TopSecret1 --cluster-identifier mycluster --encrypted --kms-key-id <kms key id>

##### Response

To verify that encryption at rest has been enabled you can confirm that the ‘Encrypted” and ‘KmsKeyId’ keys are present with values.

{

"Cluster": {

"ClusterVersion": "1.0",

"NodeType": "ds2.xlarge",

"PubliclyAccessible": true,

"Tags": [],

"MasterUsername": "adminuser",

"ClusterParameterGroups": [

{

"ParameterGroupName": "default.redshift-1.0",

"ParameterApplyStatus": "in-sync"

}

],

"Encrypted": true,

"ClusterSecurityGroups": [

{

"Status": "active",

"ClusterSecurityGroupName": "default"

}

],

"AllowVersionUpgrade": true,

"VpcSecurityGroups": [],

"KmsKeyId": <KMS Key ARN>,

"NumberOfNodes": 2,

"AutomatedSnapshotRetentionPeriod": 1,

"ClusterStatus": "creating",

"ClusterIdentifier": "mycluster",

"DBName": "dev",

"PreferredMaintenanceWindow": "wed:05:00-wed:05:30",

"PendingModifiedValues": {

"MasterUserPassword": "\*\*\*\*"

}

}

}

#### CloudTrail Event - CreateCluster

Keys and Values of interest in the CloudTrail event will be   
“eventName”: “CreateCluster”

“kmsKeyId”: The KMS Key Id being used for encryption

“encrypted”: true

{

<Omitted text>

"eventSource": "redshift.amazonaws.com",

"eventName": "CreateCluster",

"awsRegion": "us-east-1",

"sourceIPAddress": "<IP Address>",

"userAgent": "aws-cli/1.9.1 Python/2.7.10 Linux/4.1.7-15.23.amzn1.x86\_64 botocore/1.3.1",

"requestParameters": {

"clusterIdentifier": "mycluster",

"kmsKeyId": "<KMS Key Id>",

"encrypted": true,

"numberOfNodes": 2,

"masterUsername": "adminuser",

"masterUserPassword": "\*\*\*\*",

"nodeType": "ds2.xlarge"

},

"responseElements": null,

"requestID": "9f9bca22-bcf0-11e8-9d69-49043eed2cc5",

"eventID": "68dab7a9-42df-4462-8f10-f60cf651009f",

"eventType": "AwsApiCall",

"recipientAccountId": "<Account Id>"

}

#### Scenario – Launching an Amazon Redshift cluster without encryption enabled

##### CLI Command

In this scenario we are not encrypting the Redshift cluster. This means that neither the “--encrypted” or “--kms-key-id” options are used.

aws redshift create-cluster --node-type ds2.xlarge --number-of-nodes 2 --master-username adminuser --master-user-pass TopSecret1 --cluster-identifier mycluster

##### Response

The response confirms that encryption is not enabled as the “Encrypted” key has a value of “false”. There is also no “KmsKeyId” key in the response.

{

"Cluster": {

"ClusterVersion": "1.0",

"NodeType": "ds2.xlarge",

"PubliclyAccessible": true,

"Tags": [],

"MasterUsername": "adminuser",

"ClusterParameterGroups": [

{

"ParameterGroupName": "default.redshift-1.0",

"ParameterApplyStatus": "in-sync"

}

],

"Encrypted": false,

"ClusterSecurityGroups": [

{

"Status": "active",

"ClusterSecurityGroupName": "default"

}

],

"AllowVersionUpgrade": true,

"VpcSecurityGroups": [],

"NumberOfNodes": 2,

"AutomatedSnapshotRetentionPeriod": 1,

"ClusterStatus": "creating",

"ClusterIdentifier": "mycluster",

"DBName": "dev",

"PreferredMaintenanceWindow": "wed:05:00-wed:05:30",

"PendingModifiedValues": {

"MasterUserPassword": "\*\*\*\*"

}

}

}

#### CloudTrail Event - CreateCluster

Keys and Values of interest in the CloudTrail event will be “eventName”: “CreateCluster”,

“kmsKeyId”, and “encrypted”. If the “kmsKeyId” and “encrypted” keys do not exist, the Redshift cluster is not encrypted at rest.

{

<Omitted text>

"eventSource": "redshift.amazonaws.com",

"eventName": "CreateCluster",

"awsRegion": "us-east-1",

"sourceIPAddress": "<IP Address>",

"userAgent": "aws-cli/1.9.1 Python/2.7.10 Linux/4.1.7-15.23.amzn1.x86\_64 botocore/1.3.1",

"requestParameters": {

"clusterIdentifier": "mycluster",

"numberOfNodes": 2,

"masterUsername": "adminuser",

"masterUserPassword": "\*\*\*\*",

"nodeType": "ds2.xlarge"

},

"responseElements": null,

"requestID": "9f9bca22-bcf0-11e8-9d69-49043eed2cc5",

"eventID": "68dab7a9-42df-4462-8f10-f60cf651009f",

"eventType": "AwsApiCall",

"recipientAccountId": "<Account Id>"

}

##### AWS CloudFormation Snippets

When creating an Amazon Redshift cluster using AWS CloudFormation, you can declare Encryption and the KMSKeyId using the “Encrypted” and “KmsKeyId” in the “AWS::Redshift::Cluster” type.

Type: "AWS::Redshift::Cluster"

Properties:

Encrypted: Boolean

KmsKeyId: String

Encrypted

Indicates whether the data in the cluster is encrypted at rest. The default value is false.

*Required*: No

*Type*: Boolean

*Update requires*: [Replacement](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-update-behaviors.html#update-replacement)

KmsKeyId

The ID of the AWS Key Management Service (AWS KMS) key that you want to use to encrypt data in the cluster.

*Required*: No

*Type*: String

*Update requires*: [Replacement](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-update-behaviors.html#update-replacement)

### KMS Keys Should Not Be Default Key

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Ensure that no KMS Default Keys are used | Keys are created and not service defaults |

Redshift clusters can be encrypted at rest using KMS Keys. When declaring encryption, you can declare a specific KMS Key to use. If a KMS Key is not declared, a default key will be used. Some customer policy suggest default keys should not be utilized.

#### Scenario – Launching an Amazon Redshift cluster with encryption enabled and specified KMS Key

##### CLI Command

When using the AWS CLI, you can encrypt the Redshift cluster using the “--encrypted” option. This will enable KMS encryption on the Redshift cluster using a default KMS Key. Some customers policy require default KMS keys are not be used. In that case you can add the “--kms-key-id” option and provide a kms key id that can be used.

aws redshift create-cluster --node-type ds2.xlarge --number-of-nodes 2 --master-username adminuser --master-user-pass TopSecret1 --cluster-identifier mycluster --encrypted --kms-key-id <kms key id>

##### Response

To verify that encryption at rest has been enabled you can confirm that the “Encrypted” and “KmsKeyId” fields are present with values. The “KmsKeyId” key will contain the value of the KMS Key which can then be referenced to determine if the KMS Key is a default key.

{

"Cluster": {

"ClusterVersion": "1.0",

"NodeType": "ds2.xlarge",

"PubliclyAccessible": true,

"Tags": [],

"MasterUsername": "adminuser",

"ClusterParameterGroups": [

{

"ParameterGroupName": "default.redshift-1.0",

"ParameterApplyStatus": "in-sync"

}

],

"Encrypted": true,

"ClusterSecurityGroups": [

{

"Status": "active",

"ClusterSecurityGroupName": "default"

}

],

"AllowVersionUpgrade": true,

"VpcSecurityGroups": [],

"KmsKeyId": <KMS Key ARN>,

"NumberOfNodes": 2,

"AutomatedSnapshotRetentionPeriod": 1,

"ClusterStatus": "creating",

"ClusterIdentifier": "mycluster",

"DBName": "dev",

"PreferredMaintenanceWindow": "wed:05:00-wed:05:30",

"PendingModifiedValues": {

"MasterUserPassword": "\*\*\*\*"

}

}

}

#### CloudTrail Event - CreateCluster

Keys and Values of interest in the CloudTrail event will be   
“eventName”: “CreateCluster”

“kmsKeyId”: The KMS Key Id being used for encryption

“encrypted”: true

{

<Omitted text>

"eventSource": "redshift.amazonaws.com",

"eventName": "CreateCluster",

"awsRegion": "us-east-1",

"sourceIPAddress": "<IP Address>",

"userAgent": "aws-cli/1.9.1 Python/2.7.10 Linux/4.1.7-15.23.amzn1.x86\_64 botocore/1.3.1",

"requestParameters": {

"clusterIdentifier": "mycluster",

"kmsKeyId": "<KMS Key Id>",

"encrypted": true,

"numberOfNodes": 2,

"masterUsername": "adminuser",

"masterUserPassword": "\*\*\*\*",

"nodeType": "ds2.xlarge"

},

<Omitted text>

}

#### Scenario – Launching an Amazon Redshift cluster with encryption enabled and default KMS Key

##### CLI Command

When using the AWS CLI, you can encrypt the Redshift cluster using the “--encrypted” parameter. This will enable KMS encryption on the Redshift cluster using a default KMS Key. (Some customers policy require default KMS keys are not be used. In that case you can add the ‘--kms-key-id’ option and provide a kms key id that can be used.)

aws redshift create-cluster --node-type ds2.xlarge --number-of-nodes 2 --master-username adminuser --master-user-pass TopSecret1 --cluster-identifier mycluster --encrypted

##### Response

To verify that encryption at rest has been enabled you can confirm that the “Encrypted” and “KmsKeyId” fields are present with values. The “KmsKeyId” key will contain the value of the KMS Key which can then be referenced to determine if the KMS Key is a default key.

{

"Cluster": {

"ClusterVersion": "1.0",

"NodeType": "ds2.xlarge",

"PubliclyAccessible": true,

"Tags": [],

"MasterUsername": "adminuser",

"ClusterParameterGroups": [

{

"ParameterGroupName": "default.redshift-1.0",

"ParameterApplyStatus": "in-sync"

}

],

"Encrypted": true,

"ClusterSecurityGroups": [

{

"Status": "active",

"ClusterSecurityGroupName": "default"

}

],

"AllowVersionUpgrade": true,

"VpcSecurityGroups": [],

"KmsKeyId": <KMS Key ARN>,

"NumberOfNodes": 2,

"AutomatedSnapshotRetentionPeriod": 1,

"ClusterStatus": "creating",

"ClusterIdentifier": "mycluster",

"DBName": "dev",

"PreferredMaintenanceWindow": "wed:05:00-wed:05:30",

"PendingModifiedValues": {

"MasterUserPassword": "\*\*\*\*"

}

}

}

#### CloudTrail Event - CreateCluster

Keys and Values of interest in the CloudTrail event will be “eventName”: “CreateCluster”, “kmsKeyId” and “encrypted”: true. When the “kmsKeyId” key is missing from the “requestParameters” it is an indication that the default KMS Key is being used.

{

<Omitted text>

"eventSource": "redshift.amazonaws.com",

"eventName": "CreateCluster",

"awsRegion": "us-east-1",

"sourceIPAddress": "<IP Address>",

"userAgent": "aws-cli/1.9.1 Python/2.7.10 Linux/4.1.7-15.23.amzn1.x86\_64 botocore/1.3.1",

"requestParameters": {

"clusterIdentifier": "mycluster",

"encrypted": true,

"numberOfNodes": 2,

"masterUsername": "adminuser",

"masterUserPassword": "\*\*\*\*",

"nodeType": "ds2.xlarge"

},

<Omitted text>

}

#### Scenario - Determining if a KMS Key Id is the default key

You can determine if the default KMS Key is being used by issuing the DescribeKey API call with the KMS Key Id or ARN. This will return metadata regarding the KMS Key. The ‘Origin’ field reveals the source of the key material. When this value is “AWS\_KMS” , the KMS service created the key material. When this value is EXTERNAL , the key material was imported from your existing key management infrastructure or the CMK lacks key material. An additional field, “KeyManager”, reveals the key’s manager. CMKs are either customer managed or AWS managed. An AWS Managed key signifies that the key is a default KMS Key. Default KMS Keys will also have an “Origin” of “AWS\_KMS”.

##### AWS CLI Command

Aws kms describe-key --key-id <KMS Key Id or Arn>

##### Response

{

"KeyMetadata": {

"Origin": "AWS\_KMS",

"KeyId": "<KMS Key Id>",

"Description": "Default master key that protects my Redshift clusters when no other key is defined",

"KeyManager": "AWS",

"Enabled": true,

"KeyUsage": "ENCRYPT\_DECRYPT",

"KeyState": "Enabled",

"CreationDate": 1537459703.564,

"Arn": "<KMS Key Arn>",

"AWSAccountId": "<Account Id>"

}

}

##### AWS CloudFormation Snippets

When creating an Amazon Redshift cluster using AWS CloudFormation, you can declare Encryption and the KMSKeyId using the ‘Encrypted’ and ‘KmsKeyId’ in the ‘AWS::Redshift::Cluster’ type. If the ‘KmsKeyId’ parameter is not used, the default KMS Key will be used.

Type: "AWS::Redshift::Cluster"

Properties:

Encrypted: Boolean

KmsKeyId: String

Encrypted

Indicates whether the data in the cluster is encrypted at rest. The default value is false.

*Required*: No

*Type*: Boolean

*Update requires*: [Replacement](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-update-behaviors.html#update-replacement)

KmsKeyId

The ID of the AWS Key Management Service (AWS KMS) key that you want to use to encrypt data in the cluster.

*Required*: No

*Type*: String

*Update requires*: [Replacement](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-update-behaviors.html#update-replacement)

### KMS Keys Must Use Imported Key Material

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Ensure that KMS Keys used for encryption on Redshift cluster tagged as Critical or Highly Sensitive are from imported key material | I can prove that generated key material is from customer. |

Redshift clusters can be encrypted at rest using KMS Keys. When declaring encryption, you can declare a specific KMS Key to use. Some customer policy suggest Redshift clusters that are tagged as ‘Critical’ or ‘Highly Sensitive’ should not use a managed KMS Key but should have Key Data Imported. KMS has an ‘Origin’ parameter that will have a value of ‘EXTERNAL’ when key material is imported.

Information on how to create a KMS Key from imported material can be found in the following KMS documentation.

<https://docs.aws.amazon.com/kms/latest/developerguide/importing-keys.html>

#### Scenario – Launching an Amazon Redshift cluster with encryption enabled and imported KMS Key material

##### CLI Command

When using the AWS CLI, you can encrypt the Redshift cluster using the “--encrypted” option. This will enable KMS encryption on the Redshift cluster using a default KMS Key. Some customer policy require default KMS keys are not be used. In that case you can add the “--kms-key-id” parameter and provide a kms key id that can be used.

aws redshift create-cluster --node-type ds2.xlarge --number-of-nodes 2 --master-username adminuser --master-user-pass TopSecret1 --cluster-identifier mycluster --encrypted --kms-key-id <kms key id>

##### Response

To verify that encryption at rest has been enabled you can confirm that the “Encrypted” and “KmsKeyId” fields are present with values. The “KmsKeyId” key will contain the value of the KMS Key which can then be referenced to determine if the KMS Key is a default key.

{

"Cluster": {

"ClusterVersion": "1.0",

"NodeType": "ds2.xlarge",

"PubliclyAccessible": true,

"Tags": [],

"MasterUsername": "adminuser",

"ClusterParameterGroups": [

{

"ParameterGroupName": "default.redshift-1.0",

"ParameterApplyStatus": "in-sync"

}

],

"Encrypted": true,

"ClusterSecurityGroups": [

{

"Status": "active",

"ClusterSecurityGroupName": "default"

}

],

"AllowVersionUpgrade": true,

"VpcSecurityGroups": [],

"KmsKeyId": <KMS Key ARN>,

"NumberOfNodes": 2,

"AutomatedSnapshotRetentionPeriod": 1,

"ClusterStatus": "creating",

"ClusterIdentifier": "mycluster",

"DBName": "dev",

"PreferredMaintenanceWindow": "wed:05:00-wed:05:30",

"PendingModifiedValues": {

"MasterUserPassword": "\*\*\*\*"

}

}

}

#### CloudTrail Event - CreateCluster

Keys and Values of interest in the CloudTrail event will be   
“eventName”: “CreateCluster”

“kmsKeyId”: The KMS Key Id being used for encryption

“encrypted”: true

{

<Omitted text>

"eventSource": "redshift.amazonaws.com",

"eventName": "CreateCluster",

"awsRegion": "us-east-1",

"sourceIPAddress": "<IP Address>",

"userAgent": "aws-cli/1.9.1 Python/2.7.10 Linux/4.1.7-15.23.amzn1.x86\_64 botocore/1.3.1",

"requestParameters": {

"clusterIdentifier": "mycluster",

"kmsKeyId": "<KMS Key Id>",

"encrypted": true,

"numberOfNodes": 2,

"masterUsername": "adminuser",

"masterUserPassword": "\*\*\*\*",

"nodeType": "ds2.xlarge"

},

<Omitted text>

}

#### Scenario - Determining if a KMS Key has Imported Key Material

You can determine if imported key material is being used by issuing the DescribeKey API call with the KMS Key Id or ARN. This will return metadata regarding the KMS Key. The “Origin” field reveals the source of the key material. When this value is “AWS\_KMS” , the KMS service created the key material. When this value is “EXTERNAL” , the key material was imported from your existing key management infrastructure or the CMK lacks key material.

##### AWS CLI Command

Aws kms describe-key --key-id <KMS Key Id or Arn>

##### Response

{

"KeyMetadata": {

"Origin": "EXTERNAL",

"KeyId": "<KMS Key Id>",

"Description": "Imported Key Material for Redshift administration",

"KeyManager": "CUSTOMER",

"ExpirationModel": "KEY\_MATERIAL\_DOES\_NOT\_EXPIRE",

"Enabled": true,

"KeyUsage": "ENCRYPT\_DECRYPT",

"KeyState": "Enabled",

"CreationDate": 1511278571.179,

"Arn": "<KMS Key ARN>",

"AWSAccountId": "<Account Id>"

}

}

##### AWS CloudFormation Snippets

When creating an Amazon Redshift cluster using AWS CloudFormation, you can declare Encryption and the KMSKeyId using the ‘Encrypted’ and ‘KmsKeyId’ in the ‘AWS::Redshift::Cluster’ type. If the ‘KmsKeyId’ parameter is not used, the default KMS Key will be used.

Type: "AWS::Redshift::Cluster"

Properties:

Encrypted: Boolean

KmsKeyId: String

Encrypted

Indicates whether the data in the cluster is encrypted at rest. The default value is false.

*Required*: No

*Type*: Boolean

*Update requires*: [Replacement](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-update-behaviors.html#update-replacement)

KmsKeyId

The ID of the AWS Key Management Service (AWS KMS) key that you want to use to encrypt data in the cluster.

*Required*: No

*Type*: String

*Update requires*: [Replacement](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-update-behaviors.html#update-replacement)

## Encryption-in-transit

Amazon Redshift supports Secure Sockets Layer (SSL) connections to encrypt data and server certificates to validate the server certificate that the client connects to. To support SSL connections, Amazon Redshift creates and installs an AWS Certificate Manager (ACM) issued SSL certificate on each cluster. The set of Certificate Authorities that you must trust in order to properly support SSL connections can be found at <https://s3.amazonaws.com/redshift-downloads/redshift-ca-bundle.crt>.

By default, JDBC Amazon Redshift drivers use SSL. ODBC DSNs contain an “sslmode” setting that determines how to handle encryption for client connections and server certificate verification.   
SSL provides one layer of security by encrypting data that moves between your client and cluster. Using a server certificate provides an extra layer of security by validating that the cluster is an Amazon Redshift cluster. It does so by checking the server certificate that is automatically installed on all clusters that you provision.

### SSL Should Be Enabled

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Confirm that SSL is enabled | I can verify that data is encrypted in transit |

By default, cluster databases accept a connection whether it uses SSL or not. To configure your cluster to require an SSL connection, set the “require\_SSL” parameter to true in the parameter group that is associated with the cluster.

#### Scenario - Enable SSL on a Redshift cluster

##### CLI Command

To enable SSL on a Redshift cluster you modify the cluster parameter group that is associated with the cluster. The parameter “require\_ssl” is then set to “true”.

aws redshift modify-cluster-parameter-group --parameter-group-name test --parameters ParameterName=require\_ssl,ParameterValue=true

#### Scenario - Verifying SSL is enabled on a Redshift cluster

##### CLI Command

To verify that SSL is enabled on the Redshift cluster you issue “describe-cluster-parameters” and input the parameter group name that is associated with the cluster. This will return all the parameter values, the “ParameterName”: “require\_ssl” will need a value of “true”.

aws redshift describe-cluster-parameters --parameter-group-name test

{

"Parameters": [

<omitted text>

{

"Description": "require ssl for all databaseconnections",

"DataType": "boolean",

"IsModifiable": true,

"AllowedValues": "true,false",

"Source": "user",

"ParameterValue": "true",

"ParameterName": "require\_ssl",

"ApplyType": "static"

},

<omitted text>

]

}

#### CloudTrail Event - DescribeClusterParameters

Keys and Values of interest in the CloudTrail event will be   
“eventName”: “DescribeClusterParameters”

“parameterName”: “require\_ssl”

“parameterValue”: “true”

{

<Omitted text>

"eventSource": "redshift.amazonaws.com",

"eventName": "DescribeClusterParameters",

"awsRegion": "us-east-1",

"sourceIPAddress": "<IP Address>",

"userAgent": "aws-cli/1.15.27 Python/2.7.13 Darwin/16.7.0 botocore/1.10.27",

"requestParameters": {

"parameterGroupName": "test"

},

"responseElements": {

"parameters": [

<Omitted text>

{

"source": "user",

"dataType": "boolean",

"isModifiable": true,

"parameterValue": "true",

"parameterName": "require\_ssl",

"description": "require ssl for all databaseconnections",

"applyType": "static",

"allowedValues": "true,false"

},

<Omitted text>

#### CloudTrail Event - ModifyClusterParameterGroup

Keys and Values of interest in the CloudTrail event will be   
“eventName”: “ModifyClusterParameterGroup”

“parameterName”: “require\_ssl”

“parameterValue”: “true”

{

<Omitted text>

"eventSource": "redshift.amazonaws.com",

"eventName": "ModifyClusterParameterGroup",

"awsRegion": "us-east-1",

"sourceIPAddress": "<IP Address>",

"userAgent": "aws-cli/1.15.27 Python/2.7.13 Darwin/16.7.0 botocore/1.10.27",

"requestParameters": {

"parameters": [

{

"isModifiable": false,

"parameterValue": "true",

"parameterName": "require\_ssl"

}

],

"parameterGroupName": "test"

},

"responseElements": {

"parameterGroupName": "test",

"parameterGroupStatus": "Your parameter group has been updated. If you changed only dynamic parameters, associated clusters are being modified now. If you changed static parameters, all updates, including dynamic parameters, will be applied when you reboot the associated clusters."

},

<Omitted text>

##### AWS CloudFormation Snippets

In CloudFormation you can create a cluster parameter group with SSL settings enabled in Parameters property.

myClusterParameterGroup:

Type: "AWS::Redshift::ClusterParameterGroup"

Properties:

Description: "My parameter group"

ParameterGroupFamily: "redshift-1.0"

Parameters:

-

ParameterName: "require\_ssl"

ParameterValue: "true"

Parameters

A list of parameter names and values that are allowed by the Amazon Redshift engine version that you specified in the ParameterGroupFamily property. For more information, see [Amazon Redshift Parameter Groups](https://docs.aws.amazon.com/redshift/latest/mgmt/working-with-parameter-groups.html) in the *Amazon Redshift Cluster Management Guide*.

*Required*: No

*Type*: [Amazon Redshift Parameter Type](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-property-redshift-clusterparametergroup-parameter.html)

*Update requires*: [No interruption](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-update-behaviors.html#update-no-interrupt)

### AWS Certificate Manager Should Have Imported customer Key Material

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Make sure that Amazon Certificate Manager (ACM) uses customer imported certificates | I can verify that customer generated the key material |

AWS Certificate Manager (ACM) handles the complexity of creating and managing public SSL/TLS certificates for your AWS based websites and applications. You can use public certificates provided by ACM (ACM certificates) or certificates that you import into ACM. ACM certificates can secure multiple domain names and multiple names within a domain. You can also use ACM to create wildcard SSL certificates that can protect an unlimited number of subdomains.

To import a self–signed SSL/TLS certificate into ACM, you must provide the certificate and its private key. To import a signed certificate, you must also include the certificate chain. Your certificate must satisfy the following criteria:

The certificate must specify an algorithm and key size. Currently, the following public key algorithms are supported by ACM:

* + 1024-bit RSA (RSA\_1024)
  + 2048-bit RSA (RSA\_2048)
  + 4096-bit RSA (RSA\_4096)
  + Elliptic Prime Curve 256 bit (EC\_prime256v1)
  + Elliptic Prime Curve 384 bit (EC\_secp384r1)
  + Elliptic Prime Curve 521 bit (EC\_secp521r1)

#### Scenario - Importing a certificate in to AWS Certificate Manager

##### CLI Command

The following example shows how to import a certificate using the AWS Command Line Interface (AWS CLI). The example assumes the following:

The PEM-encoded certificate is stored in a file named Certificate.pem.

The PEM-encoded certificate chain is stored in a file named CertificateChain.pem.

The PEM-encoded, unencrypted private key is stored in a file named PrivateKey.pem.

aws acm import-certificate --certificate file://Certificate.pem --certificate-chain file://CertificateChain.pem --private-key file://PrivateKey.pem

##### Response

{

"CertificateArn": "<Certificate ARN>"

}

#### Scenario - Verifying that a certificate was Imported in to AWS Certificate Manager

##### CLI Command

The following example shows how to verify that a certificate was imported in to AWS Certificate Manager using the AWS Command Line Interface (AWS CLI).

aws acm describe-certificate --certificate-arn <Certificate ARN>

##### Response

In the response the “Type” key will have a value of “IMPORTED” on imported certificates. In addition, an “ImportedAt” Key only exists on imported certificates.

{

"Certificate": {

"CertificateArn": "<Certificate ARN>",

"Status": "ISSUED",

"Options": {

"CertificateTransparencyLoggingPreference": "DISABLED"

},

"SubjectAlternativeNames": [

"example.com"

],

"DomainName": "example.com",

"NotBefore": 1537546911.0,

"RenewalEligibility": "INELIGIBLE",

"NotAfter": 1852906911.0,

"ImportedAt": 1537546969.0,

"InUseBy": [],

"SignatureAlgorithm": "SHA1WITHRSA",

"KeyAlgorithm": "RSA-2048",

"KeyUsages": [

{

"Name": "ANY"

}

],

"Serial": "ff:fc:08:77:98:7e:65:1c",

"Issuer": "Internet Widgits Pty Ltd",

"Type": "IMPORTED",

"ExtendedKeyUsages": [],

"DomainValidationOptions": [

{

"ValidationMethod": "EMAIL",

"DomainName": "example.com"

}

],

"Subject": "C=US,ST=Texas,L=Dallas,O=Internet Widgits Pty Ltd,CN=example.com,E=example@example.com"

}

}

#### CloudTrail Event - ImportCertificate

Keys and Values of interest in the CloudTrail event will be “eventName”: “Import Certificate”. Additional fields such as “privateKey” and “certificate” can be found, but the values will be random numbers.

{

<Omitted text>

"eventSource": "acm.amazonaws.com",

"eventName": "ImportCertificate",

"awsRegion": "us-east-1",

"sourceIPAddress": "<IP Address>",

"userAgent": "aws-cli/1.15.27 Python/2.7.13 Darwin/16.7.0 botocore/1.10.27",

"requestParameters": {

"privateKey": {

"hb": [<random values>],

"offset": 0,

"isReadOnly": false,

"bigEndian": true,

"nativeByteOrder": false,

"mark": -1,

"position": 0,

"limit": 32,

"capacity": 32,

"address": 0

},

"certificate": {

"hb": [<random values>],

"offset": 0,

"isReadOnly": false,

"bigEndian": true,

"nativeByteOrder": false,

"mark": -1,

"position": 0,

"limit": 1419,

"capacity": 1419,

"address": 0

}

},

"responseElements": {

"certificateArn": "<Certificate ARN>"

},

"requestID": "9450a5cb-bdba-11e8-ab6a-574b07a0619d",

"eventID": "cc1208df-15d3-43d2-9b29-a706afaabb9c",

"eventType": "AwsApiCall",

"recipientAccountId": "<Account Id>"

}

### Parameter “use\_fips\_ssl” Set To “true”

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Confirm that parameter “use\_fips\_ssl” is set to “true” | I can verify that my SSL mode is FIPS compliant |

Amazon Redshift supports an SSL mode that is compliant with Federal Information Processing Standard (FIPS) 140-2. FIPS-compliant SSL mode is disabled by default. To enable FIPS-compliant SSL mode, set both the ‘use\_fips\_ssl’ parameter and the ‘require\_SSL’ parameter to true in the parameter group that is associated with the cluster.

#### Scenario - Enabling FIPS compliant SSL on a Redshift cluster

##### CLI Command

To enable FIPS compliant SSL on a Redshift cluster you modify the cluster parameter group that is associated with the cluster. The parameter “require\_ssl” is then set to “true”.

aws redshift modify-cluster-parameter-group --parameter-group-name test --parameters '{"ParameterName": "require\_ssl","ParameterValue": "true","ParameterName": "use\_fips\_ssl","ParameterValue": "true"}'

#### Scenario - Verifying FIPS SSL is enabled on a Redshift cluster

##### CLI Command

To verify that SSL is enabled on the Redshift cluster you issue “describe-cluster-parameters” and input the parameter group name that is associated with the cluster. This will return all the parameter values, the “ParameterName”: “require\_ssl” and “use\_fips\_ssl” will both need a value of “true”.

aws redshift describe-cluster-parameters --parameter-group-name test

{

"Parameters": [

<Omitted text>

{

"Description": "require ssl for all databaseconnections",

"DataType": "boolean",

"IsModifiable": true,

"AllowedValues": "true,false",

"Source": "user",

"ParameterValue": "true",

"ParameterName": "require\_ssl",

"ApplyType": "static"

},

<Omitted text>

{

"Description": "Use fips ssl library",

"DataType": "boolean",

"IsModifiable": true,

"AllowedValues": "true,false",

"Source": "user",

"ParameterValue": "true",

"ParameterName": "use\_fips\_ssl",

"ApplyType": "static"

},

<Omitted text>

]

}

#### CloudTrail Event - DescribeClusterParameters

Keys and Values of interest in the CloudTrail event will be   
“eventName”: “DescribeClusterParameters”

“parameterName”: “require\_ssl”

“parameterValue”: “true”

“parameterName”: “use\_fips\_ssl”

“parameterValue”: “true”

{

<Omitted text>

"eventSource": "redshift.amazonaws.com",

"eventName": "DescribeClusterParameters",

"awsRegion": "us-east-1",

"sourceIPAddress": "<IP Address>",

"userAgent": "aws-cli/1.15.27 Python/2.7.13 Darwin/16.7.0 botocore/1.10.27",

"requestParameters": {

"parameterGroupName": "test"

},

"responseElements": {

"parameters": [

<Omitted text>

{

"source": "user",

"dataType": "boolean",

"isModifiable": true,

"parameterValue": "true",

"parameterName": "require\_ssl",

"description": "require ssl for all databaseconnections",

"applyType": "static",

"allowedValues": "true,false"

},

<Omitted text>

{

"source": "engine-default",

"dataType": "boolean",

"isModifiable": true,

"parameterValue": "false",

"parameterName": "use\_fips\_ssl",

"description": "Use fips ssl library",

"applyType": "static",

"allowedValues": "true,false"

},

<Omitted text>

}

#### CloudTrail Event - ModifyClusterParameterGroup

Keys and Values of interest in the CloudTrail event will be   
“eventName”: “ModifyClusterParameterGroup”

“parameterName”: “use\_fips\_ssl”

“parameterValue”: “true”

{

<Omitted text>

"eventSource": "redshift.amazonaws.com",

"eventName": "ModifyClusterParameterGroup",

"awsRegion": "us-east-1",

"sourceIPAddress": "IP Address",

"userAgent": "aws-cli/1.15.27 Python/2.7.13 Darwin/16.7.0 botocore/1.10.27",

"requestParameters": {

"parameters": [

{

"isModifiable": false,

"parameterValue": "true",

"parameterName": "use\_fips\_ssl"

}

],

"parameterGroupName": "test"

},

"responseElements": {

"parameterGroupName": "test",

"parameterGroupStatus": "Your parameter group has been updated. If you changed only dynamic parameters, associated clusters are being modified now. If you changed static parameters, all updates, including dynamic parameters, will be applied when you reboot the associated clusters."

},

"requestID": "2aef6e6e-bdb3-11e8-95ad-45ef1f352884",

"eventID": "953f7743-5d6d-41da-8132-fb1ab6ef693e",

"eventType": "AwsApiCall",

"recipientAccountId": "<Account Id>"

}

##### AWS CloudFormation Snippets

In CloudFormation you can create a cluster parameter group with SSL settings enabled in Parameters property.

myClusterParameterGroup:

Type: "AWS::Redshift::ClusterParameterGroup"

Properties:

Description: "My parameter group"

ParameterGroupFamily: "redshift-1.0"

Parameters:

-

ParameterName: "require\_ssl"

ParameterValue: "true"

-

ParameterName: "use\_fips\_ssl"

ParameterValue: "true"

Parameters

A list of parameter names and values that are allowed by the Amazon Redshift engine version that you specified in the ParameterGroupFamily property. For more information, see [Amazon Redshift Parameter Groups](https://docs.aws.amazon.com/redshift/latest/mgmt/working-with-parameter-groups.html) in the *Amazon Redshift Cluster Management Guide*.

*Required*: No

*Type*: [Amazon Redshift Parameter Type](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-property-redshift-clusterparametergroup-parameter.html)

*Update requires*: [No interruption](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-update-behaviors.html#update-no-interrupt)

### Parameter “require\_SSL” Set To “true”

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Confirm that parameter “require\_SSL” is set to “true” | I can verify that data is encrypted in transit |

By default, cluster databases accept a connection whether it uses SSL or not. To configure your cluster to require an SSL connection, set the require\_SSL parameter to true in the parameter group that is associated with the cluster.

#### Scenario - Enabling SSL on a Redshift cluster

##### CLI Command

To enable SSL on a Redshift cluster you modify the cluster parameter group that is associated with the cluster. The parameter “require\_ssl” is then set to “true”.

aws redshift modify-cluster-parameter-group --parameter-group-name test --parameters ParameterName=require\_ssl,ParameterValue=true

#### CloudTrail Event - ModifyClusterParamterGroup

Keys and Values of interest in the CloudTrail event will be   
“eventName”: “ModifyClusterParameterGroup”

“parameterName”: “require\_ssl”

“parameterValue”: “true”

{

<Omitted text>

"eventSource": "redshift.amazonaws.com",

"eventName": "ModifyClusterParameterGroup",

"awsRegion": "us-east-1",

"sourceIPAddress": "<IP Address>",

"userAgent": "aws-cli/1.15.27 Python/2.7.13 Darwin/16.7.0 botocore/1.10.27",

"requestParameters": {

"parameters": [

{

"isModifiable": false,

"parameterValue": "true",

"parameterName": "require\_ssl"

}

],

"parameterGroupName": "test"

},

"responseElements": {

"parameterGroupName": "test",

"parameterGroupStatus": "Your parameter group has been updated. If you changed only dynamic parameters, associated clusters are being modified now. If you changed static parameters, all updates, including dynamic parameters, will be applied when you reboot the associated clusters."

},

<Omitted text>

}

##### AWS CloudFormation Snippets

In CloudFormation you can create a cluster parameter group with SSL settings enabled in Parameters property.

myClusterParameterGroup:

Type: "AWS::Redshift::ClusterParameterGroup"

Properties:

Description: "My parameter group"

ParameterGroupFamily: "redshift-1.0"

Parameters:

-

ParameterName: "require\_ssl"

ParameterValue: "true"

Parameters

A list of parameter names and values that are allowed by the Amazon Redshift engine version that you specified in the ParameterGroupFamily property. For more information, see [Amazon Redshift Parameter Groups](https://docs.aws.amazon.com/redshift/latest/mgmt/working-with-parameter-groups.html) in the *Amazon Redshift Cluster Management Guide*.

*Required*: No

*Type*: [Amazon Redshift Parameter Type](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-property-redshift-clusterparametergroup-parameter.html)

*Update requires*: [No interruption](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-update-behaviors.html#update-no-interrupt)

### Certificate Chain of Trust Should Only Include Whitelisted Authorities

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Confirm that the Certificate chain of trust only includes authorized authorities | I can verify that no unauthorized authorities are in the chain of trust |

The Certificate Authority on ACM certificates can be retrieved using the AWS CLI and the describe-certificate command and finding the value of the “Issuer” key. CloudTrail will log events of DescribeCertificate but does not output an “Issuer” key. An automated approach to confirming the Issuer is an authorized authority would be to monitor CloudTrail Events for ImportCertificate and RequestCertificate. These CloudTrail Events will contain the certificate ARN which can then be described and matched against a whitelist of authorized authorities.

#### Scenario - Verifying the Certificate Authority on a Certificate

##### CLI Command

To retrieve the name of the Certificate Authority that issued the certificate you can issue the describe-certificate command.

aws acm describe-certificate --certificate-arn <Certificate ARN>

##### Response

In the response the “Issuer” key will contain the value of the Certificate Authority.

{

"Certificate": {

"CertificateArn": "<Certificate ARN>",

"Status": "ISSUED",

"Options": {

"CertificateTransparencyLoggingPreference": "DISABLED"

},

"SubjectAlternativeNames": [

"example.com"

],

"DomainName": "example.com",

"NotBefore": 1537546911.0,

"RenewalEligibility": "INELIGIBLE",

"NotAfter": 1852906911.0,

"ImportedAt": 1537546969.0,

"InUseBy": [],

"SignatureAlgorithm": "SHA1WITHRSA",

"KeyAlgorithm": "RSA-2048",

"KeyUsages": [

{

"Name": "ANY"

}

],

"Serial": "ff:fc:08:77:98:7e:65:1c",

"Issuer": "Internet Widgits Pty Ltd",

"Type": "IMPORTED",

"ExtendedKeyUsages": [],

"DomainValidationOptions": [

{

"ValidationMethod": "EMAIL",

"DomainName": "example.com"

}

],

"Subject": "C=US,ST=Texas,L=Dallas,O=Internet Widgits Pty Ltd,CN=example.com,E=example@example.com"

}

}

## Encryption Key Management

### Key Rotation Policy

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Rotate KMS encryption keys. Key creation action must be captured via AWS CloudTrail | I can verify that key rotation is occurring based on customer policy. |

Cryptographic best practices discourage extensive reuse of encryption keys. To create new cryptographic material for your AWS Key Management Service (AWS KMS) customer master keys (CMKs), you can create new CMKs, and then change your applications or aliases to use the new CMKs. Or, you can enable automatic key rotation for an existing CMK.

When you enable *automatic key rotation* for a customer managed CMK, AWS KMS generates new cryptographic material for the CMK every year. AWS KMS also saves the CMK's older cryptographic material so it can be used to decrypt data that it encrypted.

Key rotation changes only the CMK's *backing key*, which is the cryptographic material that is used in encryption operations. The CMK is the same logical resource, regardless of whether or how many times its backing key changes.

Automatic key rotation is disabled by default on customer managed CMKs. When you enable (or re-enable) key rotation, AWS KMS automatically rotates the CMK 365 days after the enable date and every 365 days thereafter. Automatic key rotation is available for all customer managed CMKs with KMS-generated key material. It is not available for CMKs that have imported key material but you can rotate these CMKs manually.

When AWS KMS rotates a CMK, it writes the KMS CMK Rotation event to Amazon CloudWatch Events. You can use this event to verify that the CMK was rotated.

#### Scenario - Manually rotating a KMS Key

You might want to create a new CMK and use it in place of a current CMK instead of enabling automatic key rotation. When the new CMK has different cryptographic material than the current CMK, using the new CMK has the same effect as changing the backing key in an existing CMK. The process of replacing one CMK with another is known as *manual key rotation*. You might prefer to rotate keys manually so you can control the rotation frequency. It's also a good solution for CMKs that are not eligible for automatic key rotation, such as CMKs with imported key material.

Because the new CMK is a different resource from the current CMK, it has a different key ID and ARN. When you change CMKs, you need to update references to the CMK ID or ARN in your applications. Aliases, which associate a friendly name with a CMK, make this process easier. Use an alias to refer to a CMK in your applications. Then, when you want to change the CMK that the application uses, change the target CMK of the alias.

##### AWS CLI Command - list-aliases

To update the target CMK of an alias, use UpdateAlias operation in the AWS KMS API. For example, this command updates the TestCMK alias to point to a new CMK. Because the operation does not return any output, the example uses the ListAliases operation to show that the alias is now associated with a different CMK.

aws kms list-aliases

{

"Aliases": [

{

"AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/TestCMK",

"AliasName": "alias/TestCMK",

"TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"

},

]

}

##### AWS CLI Command - update-alias

aws kms update-alias --alias-name TestCMK --target-key-id 0987dcba-09fe-87dc-65ba-ab0987654321

aws kms list-aliases

{

"Aliases": [

{

"AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/TestCMK",

"AliasName": "alias/TestCMK",

"TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321"

},

]

}

#### CloudTrail Event - UpdateAlias

Keys and Values of interest in the CloudTrail event will be   
“eventName”: “UpdateAlias”

“aliasName”: <KMS Key Alias>

“targetKeyId”: <KMS Key to be updated>

“ARN”: <KMS Key Alias>

“ARN”: <KMS Key Id>

{

<Omitted text>

"eventSource": "kms.amazonaws.com",

"eventName": "UpdateAlias",

"awsRegion": "us-east-1",

"sourceIPAddress": "<IP Address>",

"userAgent": "aws-cli/1.15.27 Python/2.7.13 Darwin/16.7.0 botocore/1.10.27",

"requestParameters": {

"aliasName": "<KMS Key Alias>",

"targetKeyId": "2b7e038e-d5f8-46dd-a673-e2dee680f3c9"

},

"responseElements": null,

"requestID": "0744430f-ded7-48cc-afee-df21d26dd4dc",

"eventID": "5c0df1a1-d138-4eef-a968-74e63dc9a873",

"readOnly": false,

"resources": [

{

"ARN": "<KMS Key Alias>",

"accountId": "Account Id",

"type": "AWS::KMS::Key"

},

{

"ARN": "<KMS Key Id>",

"accountId": "<Account Id>",

"type": "AWS::KMS::Key"

}

],

"eventType": "AwsApiCall",

"recipientAccountId": "<Account Id>"

}

## Infrastructure

### VPC Flow Logs Enabled on Leadernode ENI

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Confirm the VPC Flow Logs are enabled on Redshift Leader node ENI | I can capture traffic going in and out of the network interface. |

VPC Flow Logs is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your VPC. Flow log data can be published to Amazon CloudWatch Logs and Amazon S3. After you've created a flow log, you can retrieve and view its data in the chosen destination.

Flow logs can help you with a number of tasks; for example, to troubleshoot why specific traffic is not reaching an instance, which in turn helps you diagnose overly restrictive security group rules. You can also use flow logs as a security tool to monitor the traffic that is reaching your instance.

Amazon Redshift allows you to attach an Elastic IP Address to your leadernode during creation. An Elastic IP Address is associated with an Elastic Network Interface which can then have VPC flow logs enabled.

#### Scenario - Verifying that VPC Flow Logs is Enabled on Redshift leader node

There is not a direct Redshift API call that will confirm if VPC Flow Logs has been enabled on the leader node. What can be done is describing the cluster which will return a value of the public and private IP Address of the leader node. With the value of the private IP Address you can make a describe ENI call with a filter that has a value of the leader node’s private IP address. This will return the ENI of the leader node which can be used to make a describe-flow-logs API call with a filter of the leader node ENI and confirm if VPC Flow Logs is enabled.

##### AWS CLI Command - describe-clusters

In the output of the describe-cluster command you can obtain the Private IP Address of the Leader node.

aws redshift describe-clusters --cluster-identifier test

##### Response

{

"Clusters": [

{

<Omitted text>

"ClusterIdentifier": "test",

"ClusterNodes": [

{

"NodeRole": "LEADER",

"PrivateIPAddress": "<Private IP Address>",

"PublicIPAddress": "<Public IP Address>"

},

{

"NodeRole": "COMPUTE-0",

"PrivateIPAddress": "<Private IP Address>",

"PublicIPAddress": "<Public IP Address>"

},

{

"NodeRole": "COMPUTE-1",

"PrivateIPAddress": "<Private IP Address>",

"PublicIPAddress": "<Public IP Address>"

}

],

<Omitted text>

}

]

}

##### CLI Command - describe-network-interfaces

When making the describe-network-interfaces command, you can add a filter with the value of the private IP address. This will contain a value of the ENI under the key “NetworkInterfaceId”

aws ec2 describe-network-interfaces --filter Name=addresses.private-ip-address,Values=172.31.2.144

##### Response

{

"NetworkInterfaces": [

{

"Status": "in-use",

"MacAddress": "0a:bb:ef:d7:80:48",

"SourceDestCheck": true,

"AvailabilityZone": "us-east-1a",

"Description": "RedshiftNetworkInterface",

"NetworkInterfaceId": "<ENI Id>",

<Omitted text>

##### CLI Command - describe-flow-logs

With the ENI Id obtained from the describe-network-interfaces output you can call the command describe-flow-logs with a filter of the ENI Id. The output will confirm the VPC Flow Logs has been enabled on the ENI

aws ec2 describe-flow-logs --filter Name=resource-id,Values=<ENI Id>

##### Response

{

"FlowLogs": [

{

"ResourceId": "<ENI Id>",

"CreationTime": "2018-09-21T19:51:47.272Z",

"LogGroupName": "/aws/kinesisfirehose/test",

"TrafficType": "ACCEPT",

"FlowLogStatus": "ACTIVE",

"FlowLogId": "<Flow Log Id>",

"DeliverLogsPermissionArn": "<IAM Role ARN>",

"DeliverLogsStatus": "SUCCESS"

}

]

}

##### AWS CloudFormation Snippets

When creating a FlowLog you can declare the “resourced” and “resourceType

n Amazon Redshift cluster using AWS CloudFormation, you can declare Encryption and the KMSKeyId using the “Encrypted” and “KmsKeyId” in the “AWS::Redshift::Cluster” type.

Type: AWS::EC2::FlowLog

Properties:

[DeliverLogsPermissionArn](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-flowlog.html#cfn-ec2-flowlog-deliverlogspermissionarn) : *String*

[LogDestination](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-flowlog.html#cfn-ec2-flowlog-logdestination) : *String*

[LogDestinationType](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-flowlog.html#cfn-ec2-flowlog-logdestinationtype) : *String*

[LogGroupName](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-flowlog.html#cfn-ec2-flowlog-loggroupname) : *String*

[ResourceId](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-flowlog.html#cfn-ec2-flowlog-resourceid) : *String*

[ResourceType](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-flowlog.html#cfn-ec2-flowlog-resourcetype) : *String*

[TrafficType](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-flowlog.html#cfn-ec2-flowlog-traffictype) : *String*

ResourceId

The ID of the subnet, network interface, or VPC for which you want to create a flow log.

*Required*: Yes

*Type*: String

*Update requires*: [Replacement](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-update-behaviors.html#update-replacement)

ResourceType

The type of resource on which to create the flow log.

*Required*: Yes

*Type*: String

*Valid Values*: VPC | Subnet | NetworkInterface

### Leader node Security Group Does Not Contain Outbound Rules

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Confirm that the Security Group associated with the Redshift Leader Node contains no Outbound rules | No traffic can exit the leadernode |

The Leader node associated with the Redshift cluster can have an EIP and ENI attached if the setting for publicly accessible is selected. ENIs have an associated Security Group which should not contain any outbound rules.

#### Scenario - Confirm that the Leader node’s Security Group does not contain any Outbound rules

There is not a direct Redshift API call that will confirm if the Leader node’s Security Group contains any Outbound rules. What can be done is describing the cluster which will return a value of the public and private IP Address of the leader node. With the value of the private IP Address you can make a describe-network-interfaces call with a filter that has a value of the leader node’s private IP address. This will return the Security Groups associated with the ENI and can then be used as input for the describe-security-groups command.

##### AWS CLI Command - describe-clusters

In the output of the describe-cluster command you can obtain the Private IP Address of the Leader node.

aws redshift describe-clusters --cluster-identifier test3

{

"Clusters": [

{

<Omitted text>

"ClusterIdentifier": "test",

"ClusterNodes": [

{

"NodeRole": "LEADER",

"PrivateIPAddress": "<Private IP Address>",

"PublicIPAddress": "<Public IP Address>"

},

{

"NodeRole": "COMPUTE-0",

"PrivateIPAddress": "<Private IP Address>",

"PublicIPAddress": "<Public IP Address>"

},

{

"NodeRole": "COMPUTE-1",

"PrivateIPAddress": "<Private IP Address>",

"PublicIPAddress": "<Public IP Address>"

}

],

<Omitted text>

}

]

}

##### AWS CLI Command - describe-network-interfaces

When making the describe-network-interfaces command, you can add a filter with the value of the private IP address. This will contain the security groups associated with the ENI under the “Groups” key.

aws ec2 describe-network-interfaces --filter Name=addresses.private-ip-address,Values=<private IP Address>

##### Response

{

"NetworkInterfaces": [

{

<Omitted text>,

"Groups": [

{

"GroupName": "<Security Group Name>",

"GroupId": "<Security Group Id>"

},

{

"GroupName": "<Security Group Name>",

"GroupId": "<Security Group Id>"

}

],

<Omitted text>

##### AWS CLI Command - describe-security-groups

With the values of the Security Group Ids, the ec2 describe-security-groups command with the --group-ids option can be made. This will return output of the Security Group rules. The “IpPermissionsEgress” section will provide additional details regarding Outbound rules.

aws ec2 describe-security-groups --group-ids <Security Group Id>

##### Response

{

"SecurityGroups": [

{

"IpPermissionsEgress": [

{

"IpProtocol": "-1",

"PrefixListIds": [],

"IpRanges": [

{

"CidrIp": "0.0.0.0/0"

}

],

"UserIdGroupPairs": [],

"Ipv6Ranges": []

}

],

<Omitted text>

### No EIP Should Be Assigned to Redshift Cluster

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Confirm that the Redshift cluster has no EIP assigned | The cluster is not accessible outside the VPC |

#### Scenario - Redshift cluster has an EIP assigned

##### AWS CLI Command - describe-clusters

In the output of the describe-cluster command you can obtain the Public IP Addresses of the leader node or in the case of a single node cluster the Shared node.

aws redshift describe-clusters --cluster-identifier test

##### Response

{

"Clusters": [

{

<Omitted text>

"ClusterIdentifier": "test",

"ClusterNodes": [

{

"NodeRole": "LEADER",

"PrivateIPAddress": "<Private IP Address>",

"PublicIPAddress": "<Public IP Address>"

},

{

"NodeRole": "COMPUTE-0",

"PrivateIPAddress": "<Private IP Address>",

"PublicIPAddress": "<Public IP Address>"

},

{

"NodeRole": "COMPUTE-1",

"PrivateIPAddress": "<Private IP Address>",

"PublicIPAddress": "<Public IP Address>"

}

],

<Omitted text>

}

]

}

##### CLI Command - describe-address

When making the describe-addresses command, you can add a filter with the value of the public IP address. If output is provided regarding the Public IP address then the Redshift cluster has an EIP associated

aws ec2 describe-addresses --public-ips <Public IP Address>

##### Response

{

"Addresses": [

{

"Domain": "vpc",

"NetworkInterfaceId": "<ENI Id>",

"AssociationId": "<EIP Association Id>",

"NetworkInterfaceOwnerId": "<Account Id>",

"PublicIp": "<Public IP Address>",

"AllocationId": "<EIP Allocation Id>",

"PrivateIpAddress": "<Private IP Address>"

}

]

}

#### Scenario - Redshift cluster does not have an EIP assigned

##### AWS CLI Command - describe-clusters

In the output of the describe-cluster command you can obtain the Public IP Addresses of the leader node or in the case of a single node cluster the Shared node.

aws redshift describe-clusters --cluster-identifier test3

{

"Clusters": [

{

<Omitted text>

"ClusterIdentifier": "test",

"ClusterNodes": [

{

"NodeRole": "LEADER",

"PrivateIPAddress": "<Private IP Address>",

"PublicIPAddress": "<Public IP Address>"

},

{

"NodeRole": "COMPUTE-0",

"PrivateIPAddress": "<Private IP Address>",

"PublicIPAddress": "<Public IP Address>"

},

{

"NodeRole": "COMPUTE-1",

"PrivateIPAddress": "<Private IP Address>",

"PublicIPAddress": "<Public IP Address>"

}

],

<Omitted text>

}

]

}

##### CLI Command - describe-address

When making the describe-addresses command, you can add a filter with the value of the public IP address. If output is not provided regarding the Public IP address then the Redshift cluster does not have an EIP associated

aws ec2 describe-addresses --public-ips <Public IP Address>

An error occurred (InvalidAddress.NotFound) when calling the DescribeAddresses operation: Address <Public Ip Address> not found.

### VPC Endpoint Enabled for S3 Access

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Utilize VPC Endpoints for Amazon S3 access | All data to S3 remains within the private AWS network and does not traverse the public internet |

AWS PrivateLink is a VPC Endpoint which is designed for customers to access AWS services in a highly available and scalable manner, while keeping all the traffic within the AWS network. With PrivateLink, endpoints are created directly inside of your VPC, using Elastic Network Interfaces (ENIs) and IP addresses in your VPC’s subnets. The service is within your VPC, enabling connectivity to AWS services via private IP addresses. That means that VPC Security Groups can be used to manage access to the endpoints and that PrivateLink endpoints can also be accessed from your premises via AWS Direct Connect.

#### Scenario - Request to Amazon S3 Across PrivateLink Connection

##### CloudTrail Event - PutObject across PrivateLink

If the Key:Value pair of "vpcEndpointId": "<Privatelink Id>" exists, this is confirmation that the request has traversed the PrivateLink connection.

{

<Omitted text>

"eventTime": "2018-09-24T14:23:43Z",

"eventSource": "s3.amazonaws.com",

"eventName": "PutObject",

"awsRegion": "us-east-1",

"sourceIPAddress": "<IP Address>",

"userAgent": "[aws-cli/1.14.8 Python/2.7.14 Linux/4.14.62-70.117.amzn2.x86\_64 botocore/1.8.12]",

"requestParameters": {

"bucketName": "<S3 Bucket Name>",

"key": "<Key Name>"

},

"responseElements": null,

<Omitted text>

"resources": [

{

"type": "AWS::S3::Object",

"ARN": "<Object ARN>"

},

{

"accountId": "<Account Id>",

"type": "AWS::S3::Bucket",

"ARN": "<S3 Bucket ARN>"

}

],

"eventType": "AwsApiCall",

"recipientAccountId": "<Account Id>",

"vpcEndpointId": "<Privatelink Id>"

}

#### Scenario - Request to Amazon S3 with no PrivateLink connection

##### CloudTrail Event - PutObject with no PrivateLink connection

If the Key:Value pair of "vpcEndpointId": "<Privatelink Id>" does not exist, then the request has not traversed a PrivateLink connection.

{

<Omitted text>

"eventTime": "2018-09-24T14:33:37Z",

"eventSource": "s3.amazonaws.com",

"eventName": "PutObject",

"awsRegion": "us-west-2",

"sourceIPAddress": "<Public IP Address>",

"userAgent": "[aws-cli/1.9.1 Python/2.7.10 Linux/4.1.7-15.23.amzn1.x86\_64 botocore/1.3.1]",

"requestParameters": {

"bucketName": "<S3 bucketName>",

"key": "<Key Name>"

},

"responseElements": {

"x-amz-version-id": "ldZdkuuMf3hi3P64JzYYWOvZ1vVh\_JEO"

},

"additionalEventData": {

"x-amz-id-2": "+7+bZGS7wyf3UhU+cEaLjw7XConKCHubIXOwt4KW0cYi6MT6QfIGvXejoLy2+U3fc5m+oW0083Q="

},

"requestID": "2B0CEF466AE1FCB9",

"eventID": "e8915117-7244-4856-9bd7-e38b245f2428",

"readOnly": false,

"resources": [

{

"type": "AWS::S3::Object",

"ARN": "<Object ARN>"

},

{

"accountId": "<AccountId>",

"type": "AWS::S3::Bucket",

"ARN": "<S3 Bucket ARN>"

}

],

"eventType": "AwsApiCall",

"recipientAccountId": "<Account Id>"

}

## Identity and Access Management (IAM)

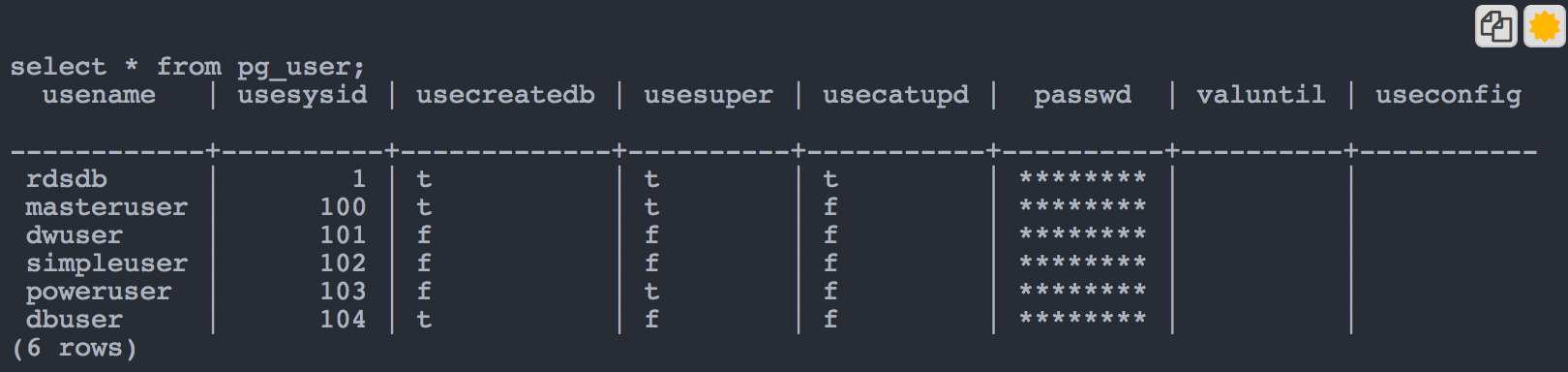
For any operation that accesses data on another AWS resource, such as using a COPY command to load data from Amazon S3, your cluster needs permission to access the resource and the data on the resource on your behalf. You provide those permissions by using AWS Identity and Access Management, either through an IAM role that is attached to your cluster or by providing the AWS access key for an IAM user that has the necessary permissions. To best protect your sensitive data and safeguard your AWS access credentials, we recommend creating an IAM role and attaching it to your cluster.

### Only Master and Superuser Local Users Exist

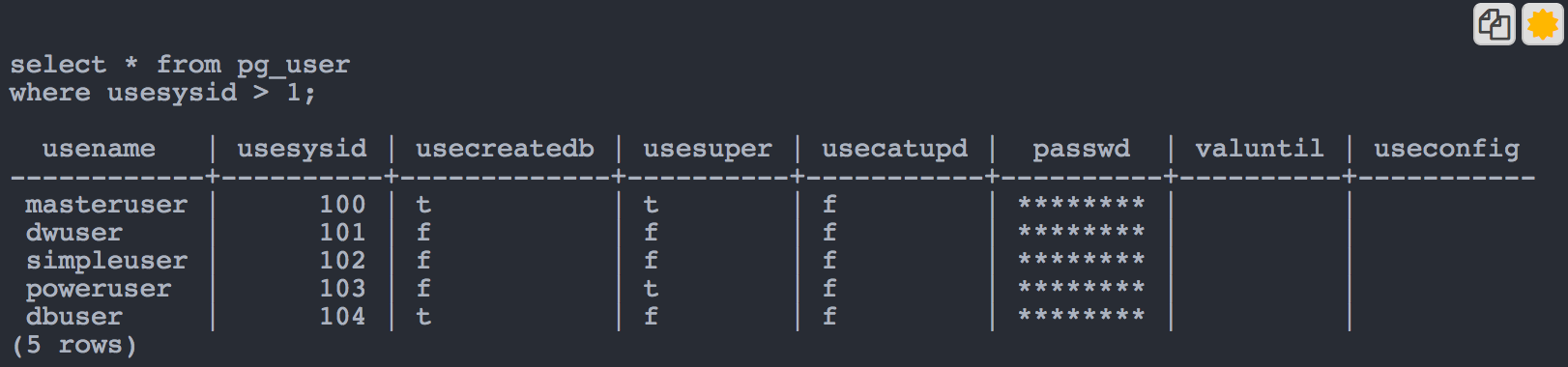
|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Confirm that only a Master or Superuser account can manage the Redshift cluster | I can verify that no unnecessary local users exist |

#### Scenario - Query the PG\_USER catalog

You can query the PG\_USER catalog to view a list of all database users, along with the user ID (USESYSID) and user privileges.



The user name rdsdb is used internally by Amazon Redshift to perform routine administrative and maintenance tasks. You can filter your query to show only user-defined user names by adding where usesysid > 1 to your select statement.



### Storing and Retrieving Passwords To Parameter Store With KMS

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Protect clear text passed for authentication of Master user using AWS Parameter Store with KMS encryption | I can verify that passwords are protected and encrypted |

AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management. You can store data such as passwords, database strings, and license codes as parameter values. You can store values as plain text or encrypted data. You can then reference values by using the unique name that you specified when you created the parameter. Data such as passwords can be encrypted using AWS KMS.

#### Scenario - Store a Password in AWS Systems Manager Parameter Store with KMS encryption

With AWS Systems Manager Parameter Store, you can create [Secure String parameters](https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-paramstore-about.html#sysman-paramstore-securestring), which are parameters that have a plaintext parameter name and an encrypted parameter value. Parameter Store uses AWS KMS to encrypt and decrypt the parameter values of Secure String parameters.

##### AWS CLI - put-parameter with KMS Key

When using the CLI to store values that you want encrypted by KMS, the following options will be used

--type SecureString

(optional)--key-id - This allows you to declare which KMS Key to encrypt and decrypt with. If this is not provided the default KMS key will be used.

aws ssm put-parameter --name redshift-user --description "Redshift User Password" --value P@$$w0rd! --type SecureString --key-id <KMS Key Id>

##### Response

{

"Version": 1

}

#### CloudTrail Event - PutParameter with KMS Key

When analyzing the PutParameter CloudTrail event, you can verify that KMS encryption is used by evaluating the “type” and “keyId” keys under the “requestParameters” key. The “type” should be “SecureString” and the “keyId” will hold the value of the KMS Key that is being used. The value of the parameter is also not part of the output of the event as it is encrypted.

{

<Omitted text >

"eventSource": "ssm.amazonaws.com",

"eventName": "PutParameter",

"awsRegion": "us-east-1",

"sourceIPAddress": "<IP Address>",

"userAgent": "aws-cli/1.14.8 Python/2.7.14 Linux/4.14.62-70.117.amzn2.x86\_64 botocore/1.8.12",

"requestParameters": {

"name": "redshift-user",

"description": "Redshift User Password",

"type": "SecureString",

"keyId": "<KMS Key Id>"

},

"responseElements": {

"version": 1

},

"requestID": "6fb4b65a-1105-4865-89ac-a0d04811193d",

"eventID": "b35f9051-10e0-4b6f-83ce-be196fe81aec",

"resources": [

{

"ARN": "<SSM Parameter ARN>",

"accountId": "<Account Id>"

}

],

"eventType": "AwsApiCall",

"recipientAccountId": "<Account Id>"

}

#### CloudTrail Event - PutParameter no Encryption

If the PutParameter does not contain a “type” of “SecureString” or “keyId” value under the “requestParameters” key. Then the SSM parameter has not been encrypted. Also, of note that the “value” field will exist with a cleartext copy of the value.

{

<Omitted text>

"eventSource": "ssm.amazonaws.com",

"eventName": "PutParameter",

"awsRegion": "us-east-1",

"sourceIPAddress": "<IP Address>",

"userAgent": "aws-cli/1.14.8 Python/2.7.14 Linux/4.14.62-70.117.amzn2.x86\_64 botocore/1.8.12",

"requestParameters": {

"name": "redshift-user2",

"description": "Redshift User Password",

"value": " P@3497w0rd!",

"type": "String"

},

"responseElements": {

"version": 1

},

"requestID": "8019952a-cb25-42b0-ac6c-585764162f24",

"eventID": "91cc81fd-e1fc-45a0-a805-cc18469d2bbf",

"resources": [

{

"ARN": "<SSM Parameter ARN>",

"accountId": "<Account Id>"

}

],

"eventType": "AwsApiCall",

"recipientAccountId": "<Account Id>"

}

### IAM Role Policies Should Include “redshift:RequestTag” Condition

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Confirm that the IAM Role Policy includes a value for the Condition key of “redshift:RequestTag” and is not blank. | I can verify that blanket authority is not assigned across all clusters |

When you grant permissions, you can use the access policy language to specify the conditions when a policy should take effect. To identify conditions where a permissions policy applies, include a Condition element in your IAM permissions policy. For example, you can create a policy that permits a user to create a cluster using the redshift:CreateCluster action, and you can add a Condition element to restrict that user to only create the cluster in a specific region. In Amazon Redshift you can use two condition keys to restrict access to resources based on the tags for those resources, redshift:RequestTag and redshift:ResourceTag.

#### Scenario - IAM Role Policy with Condition Key of “redshift:RequestTag”

The “redshift:RequestTag” requires users to include a tag key and value whenever they create a resource. The redshift:RequestTag condition key only applies to Amazon Redshift API actions that create a resource.

##### IAM Policy Example

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "RedshiftManual Policy",

"Effect": "Allow",

"Action": "redshift:\*",

"Resource": "\*",

"Condition": {

"StringEquals": {

"redshift:RequestTag/Name": [

"example-tag"

]

}

}

}

]

}

#### Scenario - Create a Redshift Cluster with redshift:ResourceTag IAM Condition Policy - Success

Using the example IAM policy provided, you can create a cluster and declare the resource tag during creation using the --tags option.

##### AWS CLI - Create-Cluster

aws redshift create-cluster --node-type ds2.xlarge --number-of-nodes 2 --master-username adminuser --master-user-pass TopSecret1 --cluster-identifier mycluster2 --tags Key=Name,Value=example-tag

##### Result - Success

{

"Cluster": {

"ClusterVersion": "1.0",

"NodeType": "ds2.xlarge",

"PubliclyAccessible": true,

"Tags": [

{

"Value": "example-tag",

"Key": "Name"

}

],

"MasterUsername": "adminuser",

<Omitted text>

#### Scenario - Create a Redshift Cluster with redshift:ResourceTag IAM Condition Policy - AccessDenied

Using the example IAM policy provided, you attempt to create a cluster with a different tag value and receive an AccessDenied response

##### AWS CLI - Create-Cluster

aws redshift create-cluster --node-type ds2.xlarge --number-of-nodes 2 --master-username adminuser --master-user-pass TopSecret1 --cluster-identifier mycluster2 --tags Key=Name,Value=any-other-value

##### Result - AccessDenied

A client error (AccessDenied) occurred when calling the CreateCluster operation: User: <IAM User> is not authorized to perform: redshift:CreateCluster on resource: <Redshift cluster ARN>

### Redshift Clusters Must Use Service Linked Role

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Ensure that Redshift clusters run with a Service Linked Role | Proper permissions are being used |

Amazon Redshift uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to Amazon Redshift. Service-linked roles are predefined by Amazon Redshift and include all the permissions that the service requires to call AWS services on behalf of your Amazon Redshift cluster.

A service-linked role makes setting up Amazon Redshift easier because you don’t have to manually add the necessary permissions. The role is linked to Amazon Redshift use cases and has predefined permissions. Only Amazon Redshift can assume the role, and only the service-linked role can use the predefined permissions policy.

Amazon Redshift creates a service-linked role in your account the first time you create a cluster. You can delete the service-linked role only after you delete all of the Amazon Redshift clusters in your account. This protects your Amazon Redshift resources because you can't inadvertently remove permissions needed for access to the resources.

#### Scenario - Service-Linked Role Permissions for Amazon Redshift

Amazon Redshift uses the service-linked role named **AWSServiceRoleForRedshift** – Allows Amazon Redshift to call AWS services on your behalf. The AWSServiceRoleForRedshift service-linked role trusts only redshift.amazonaws.com to assume the role.

The AWSServiceRoleForRedshift service-linked role permissions policy allows Amazon Redshift to complete the following on all related resources:

* ec2:DescribeVpcs
* ec2:DescribeSubnets
* ec2:DescribeNetworkInterfaces
* ec2:DescribeAddress
* ec2:AssociateAddress
* ec2:DisassociateAddress
* ec2:CreateNetworkInterface
* ec2:DeleteNetworkInterface
* ec2:ModifyNetworkInterfaceAttribute

##### IAM Role - AWSServiceRoleForRedshift Permissions

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"ec2:DescribeVpcs",

"ec2:DescribeSubnets",

"ec2:DescribeNetworkInterfaces",

"ec2:DescribeAddresses",

"ec2:AssociateAddress",

"ec2:DisassociateAddress",

"ec2:CreateNetworkInterface",

"ec2:DeleteNetworkInterface",

"ec2:ModifyNetworkInterfaceAttribute"

],

"Resource": "\*"

}

]

}

##### IAM Role - AWSServiceRoleForRedshift Trusted Entities

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": "redshift.amazonaws.com"

},

"Action": "sts:AssumeRole"

}

]

}

#### Scenario - Create a Redshift Cluster with an IAM Role attached

You can declare an IAM Role to attach to a Redshift Cluster upon creation using the --iam-role option and providing the ARN of the IAM Role to attach.

##### AWS CLI - Create-Cluster with IAM Role Attached

aws redshift create-cluster --node-type ds2.xlarge --number-of-nodes 2 --master-username adminuser --master-user-pass TopSecret1 --cluster-identifier mycluster --iam-role AWSServiceRoleForRedshift

##### Result

The result will return the “IamRoles” key which will contain the value of the IAM Roles that have been attached to the cluster.

{

"Cluster": {

"IamRoles": [

{

"ApplyStatus": "adding",

"IamRoleArn": "<IAM Role Arn>"

}

],

"ClusterVersion": "1.0",

"NodeType": "ds2.xlarge",

"PubliclyAccessible": true,

"Tags": [],

"MasterUsername": "adminuser",

<Omitted text>

#### Scenario - Create a Redshift Cluster without an IAM Role attached

##### AWS CLI - Create-Cluster with IAM Role Attached

aws redshift create-cluster --node-type ds2.xlarge --number-of-nodes 2 --master-username adminuser --master-user-pass TopSecret1 --cluster-identifier mycluster

##### Result

The result will not contain an “IamRoles” key which indicates that no IAM Role was attached during creation.

{

"Cluster": {

"ClusterVersion": "1.0",

"NodeType": "ds2.xlarge",

"PubliclyAccessible": true,

"Tags": [],

"MasterUsername": "adminuser",

<Omitted text>

#### Scenario - Describe Cluster to Retrieve IAM Roles attached to the Redshift Cluster

##### AWS CLI - Create-Cluster with IAM Role Attached

Aws redshift describe-cluster --cluster-identifier mycluster

##### Result

The result will contain a key “iamRoles” which will contain the ARN values of all the IAM Roles attached to the Redshift cluster. If the value of the “iamRoles” key is empty, this indicates that there are no IAM Roles attached.

{

"Clusters": [

{

"PubliclyAccessible": true,

"MasterUsername": "adminuser",

"VpcSecurityGroups": [],

"ClusterPublicKey": "<Omitted text>",

"NumberOfNodes": 2,

<Omitted text>

"IamRoles": [

{

"ApplyStatus": "in-sync",

"IamRoleArn": "<IAM Role Arn>"

}

],

<Omitted text>

}

]

}

##### CloudTrail Event - CreateCluster with IAM Role

Within the CloudTrail even under the “requestParameters” key there will be a sub-key “iamRoles” that will contain the values of the IAM Roles that are attached to the cluster. In addition, there will be an “iamRoles” value under the “responseElements” key.

{

<Omitted text>

"eventSource": "redshift.amazonaws.com",

"eventName": "CreateCluster",

<Omitted text>

"requestParameters": {

"clusterIdentifier": "mycluster",

"numberOfNodes": 2,

"masterUsername": "adminuser",

"masterUserPassword": "\*\*\*\*",

"iamRoles": [

"<IAM Role Arn>"

],

"nodeType": "ds2.xlarge"

},

"responseElements": {

<Omitted text>

"iamRoles": [

{

"iamRoleArn": "<IAM Role ARN>",

"applyStatus": "adding"

}

],

<Omitted text>

##### CloudTrail Event - CreateCluster without an IAM Role

Within the CloudTrail even under the “requestParameters” key there will be a sub-key “iamRoles” that will contain the values of the IAM Roles that are attached to the cluster. If the “iamRoles” key does not exist under the “requestParameters” then the cluster was created without an IAM role attached. In addition, there will be an “iamRoles” value under the “responseElements” key.

{

<Omitted text>

"eventSource": "redshift.amazonaws.com",

"eventName": "CreateCluster",

<Omitted text>

"requestParameters": {

"clusterIdentifier": "mycluster",

"numberOfNodes": 2,

"masterUsername": "adminuser",

"masterUserPassword": "\*\*\*\*",

"nodeType": "ds2.xlarge"

},

"responseElements": {

<Omitted text>

"iamRoles":[],

<Omitted text>

### Database Users Should Have Password Disabled

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Confirm that DB users have password disabled. | passwords are not stored locally |

You can create users within Amazon Redshift. When you create a user, you can choose to disable the password. You can also alter the user and disable the password at that time.

#### Scenario - Create user with password disabled

##### Redshift command - CREATE USER

When setting the user's password, you can use the PASSWORD parameter to DISABLE the password.

PASSWORD { '*password*' | '*md5hash*' | DISABLE }

By default, users can change their own passwords, unless the password is disabled. To disable a user's password, specify DISABLE. When a user's password is disabled, the password is deleted from the system and the user can log on only using temporary IAM user credentials. Only a superuser can enable or disable passwords.

CREATE USER *testuser* PASSWORD DISABLE

#### Scenario - Create user with password disabled

##### Redshift command - ALTER USER

When setting the user's password, you can use the PASSWORD parameter to DISABLE the password.

PASSWORD { '*password*' | '*md5hash*' | DISABLE }

By default, users can change their own passwords, unless the password is disabled. To disable a user's password, specify DISABLE. When a user's password is disabled, the password is deleted from the system and the user can log on only using temporary IAM user credentials. Only a superuser can enable or disable passwords.

ALTER USER *testuser* PASSWORD DISABLE

### Snapshots Copied To Another Region Are Re-tagged With New Region

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Migrate snapshots to a different region and have tags recreated to signify the region | Snapshots have the proper tagged values |

Tags are not required for resources in Amazon Redshift, but they help provide context. You might want to tag resources with metadata about cost centers, project names, and other pertinent information related to the resource. For example, suppose you want to track which resources belong to a test environment and a production environment. You could create a key named environment and provide the value test or production to identify the resources used in each environment. If you use tagging in other AWS services or have standard categories for your business, we recommend that you create the same key-value pairs for resources in Amazon Redshift for consistency.

Tags are retained for resources after you resize a cluster, and after you restore a snapshot of a cluster within the same region. However, tags are not retained if you copy a snapshot to another region, so you must recreate the tags in the new region. If you delete a resource, any associated tags are deleted.

You can configure Amazon Redshift to automatically copy snapshots (automated or manual) for a cluster to another region. When a snapshot is created in the cluster’s primary region, it will be copied to a secondary region; these are known respectively as the *source region* and *destination region*. By storing a copy of your snapshots in another region, you have the ability to restore your cluster from recent data if anything affects the primary region. You can configure your cluster to copy snapshots to only one destination region at a time.

A strategy for re-tagging snapshots copied to another region would be to trigger a custom AWS Lambda function on the CopyClusterSnapshot CloudTrail event. The Lambda function can then use the CloudTrail event to capture the snapshot id. An additional call can be made to add a tag to the snapshot id.

#### Scenario - Enabling Cross-region Snapshots

##### AWS CLI Command - enable-snapshot-copy

Enables the automatic copy of snapshots from one region to another region for a specified cluster.

--cluster-identifier (string)

The unique identifier of the source cluster to copy snapshots from.

Constraints: Must be the valid name of an existing cluster that does not already have cross-region snapshot copy enabled.

--destination-region (string)

The destination region that you want to copy snapshots to.

Constraints: Must be the name of a valid region. For more information, see Regions and Endpoints in the Amazon Web Services General Reference.

aws redshift enable-snapshot-copy --cluster-identifier mycluster --destination-region us-west-2

##### Response

The response will include the key “ClusterSnapshotCopyStatus” with sub-keys of “DestinationRegion” and the “RetentionPeriod”.

{

"Cluster": {

"PubliclyAccessible": true,

"MasterUsername": "adminuser",

"VpcSecurityGroups": [],

"NumberOfNodes": 2,

"PendingModifiedValues": {},

"ClusterVersion": "1.0",

"Tags": [],

"AutomatedSnapshotRetentionPeriod": 1,

"ClusterParameterGroups": [

{

"ParameterGroupName": "default.redshift-1.0",

"ParameterApplyStatus": "in-sync"

}

],

"DBName": "dev",

"PreferredMaintenanceWindow": "thu:06:00-thu:06:30",

"Endpoint": {

"Port": 5439,

"Address": "<Redshift endpoint>"

},

"IamRoles": [

{

"ApplyStatus": "in-sync",

"IamRoleArn": "<IAM Role ARN>"

}

],

"AllowVersionUpgrade": true,

"ClusterCreateTime": "2018-09-24T18:26:33.268Z",

"EnhancedVpcRouting": false,

"ClusterSecurityGroups": [

{

"Status": "active",

"ClusterSecurityGroupName": "default"

}

],

"ClusterIdentifier": "mycluster",

"AvailabilityZone": "us-east-1c",

"NodeType": "ds2.xlarge",

"Encrypted": false,

"ClusterSnapshotCopyStatus": {

"DestinationRegion": "us-west-2",

"RetentionPeriod": 7

},

"ClusterStatus": "available"

}

}

##### CloudTrail Event - CopyClusterSnapshot

In the CloudTrail event there will be a key for both the “sourceSnapshotIdentifier” and “targetSnapshotIdentifier”.

{

<Omitted text>

"eventTime": "2018-09-24T19:23:46Z",

"eventSource": "redshift.amazonaws.com",

"eventName": "CopyClusterSnapshot",

"awsRegion": "us-west-2",

"sourceIPAddress": "<IP Address>",

"userAgent": "aws-cli/1.9.1 Python/2.7.10 Linux/4.1.7-15.23.amzn1.x86\_64 botocore/1.3.1",

"requestParameters": {

"sourceSnapshotIdentifier": "<Source Snapshot Identifier>",

"targetSnapshotIdentifier": "<Target Snapshot Identifier>"

},

"responseElements": null,

"requestID": "5b20280e-c02f-11e8-9750-739153b1841b",

"eventID": "12f9bcd8-38bb-44cf-a570-0d66ce1e3305",

"eventType": "AwsApiCall",

"recipientAccountId": "<Account Id>"

}

## Logging and Monitoring

Amazon Redshift logs information about connections and user activities in your database. These logs help you to monitor the database for security and troubleshooting purposes, which is a process often referred to as database auditing. The logs are stored in the Amazon Simple Storage Service (Amazon S3) buckets for convenient access with data security features for users who are responsible for monitoring activities in the database.

Amazon Redshift logs information in the following log files:

* *Connection log* — logs authentication attempts, and connections and disconnections.
* *User log* — logs information about changes to database user definitions.
* *User activity log* — logs each query before it is run on the database.

The connection and user logs are useful primarily for security purposes. You can use the connection log to monitor information about the users who are connecting to the database and the related connection information, such as their IP address, when they made the request, what type of authentication they used, and so on. You can use the user log to monitor changes to the definitions of database users.

The user activity log is useful primarily for troubleshooting purposes. It tracks information about the types of queries that both the users and the system perform in the database.

The connection log and user log both correspond to information that is stored in the system tables in your database. You can use the system tables to obtain the same information, but the log files provide an easier mechanism for retrieval and review. The log files rely on Amazon S3 permissions rather than database permissions to perform queries against the tables. Additionally, by viewing the information in log files rather than querying the system tables, you reduce any impact of interacting with the database.

### Confirm “enable\_user\_activity\_logging” Is Enabled and Logs Are Delivered To S3

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Confirm “enable\_user\_activity\_logging” has been enabled and that logs are being delivered to an Amazon S3 Bucket | Log user activity in the Redshift cluster |

The enable\_user\_activity\_logging parameter is disabled (false) by default, but you can set it to true to enable the user activity log.

When you enable logging, Amazon Redshift collects logging information and uploads it to log files stored in Amazon S3. You can use an existing bucket or a new bucket. Amazon Redshift requires the following IAM permissions to the bucket:

* *s3:GetBucketAcl* The service requires read permissions to the Amazon S3 bucket so it can identify the bucket owner.
* *s3:PutObject* The service requires put object permissions to upload the logs. Each time logs are uploaded, the service determines whether the current bucket owner matches the bucket owner at the time logging was enabled. If these owners do not match, logging is still enabled but no log files can be uploaded until you select a different bucket.

If you want to use a new bucket, and have Amazon Redshift create it for you as part of the configuration process, the correct permissions will be applied to the bucket. However, if you create your own bucket in Amazon S3 or use an existing bucket, you need to add a bucket policy that includes the bucket name, and the Amazon Redshift Account ID that corresponds to your region from the following table:

|  |  |  |
| --- | --- | --- |
| **Region Name** | **Region** | **Account ID** |
| US East (N. Virginia) Region | us-east-1 | 193672423079 |
| US East (Ohio) Region | us-east-2 | 391106570357 |
| US West (N. California) Region | us-west-1 | 262260360010 |
| US West (Oregon) Region | us-west-2 | 902366379725 |
| Asia Pacific (Mumbai) Region | ap-south-1 | 865932855811 |
| Asia Pacific (Seoul) Region | ap-northeast-2 | 760740231472 |
| Asia Pacific (Singapore) Region | ap-southeast-1 | 361669875840 |
| Asia Pacific (Sydney) Region | ap-southeast-2 | 762762565011 |
| Asia Pacific (Tokyo) Region | ap-northeast-1 | 404641285394 |
| China (Ningxia) Region | cn-northwest-1 | 660998842044 |
| Canada (Central) Region | ca-central-1 | 907379612154 |
| EU (Frankfurt) Region | eu-central-1 | 053454850223 |
| EU (Ireland) Region | eu-west-1 | 210876761215 |
| EU (London) Region | eu-west-2 | 307160386991 |
| EU (Paris) Region | eu-west-3 | 915173422425 |
| South America (São Paulo) Region | sa-east-1 | 075028567923 |

#### Scenario - Enable the parameter “enable\_user\_activity\_logging”

##### AWS CLI - modify-cluster-parameter-group

aws redshift modify-cluster-parameter-group --parameter-group-name my-parameter-group --parameters ParameterName=enable\_user\_activity\_logging,ParameterValue=True

##### Response

{

"ParameterGroupStatus": "Your parameter group has been updated. If you changed only dynamic parameters, associated clusters are being modified now. If you changed static parameters, all updates, including dynamic parameters, will be applied when you reboot the associated clusters.",

"ParameterGroupName": "my-parameter-group"

}

#### Scenario - Verifying that the parameter “enable\_user\_activity\_logging”

##### AWS CLI - describe-cluster-parameters

aws redshift describe-cluster-parameters --parameter-group-name my-parameter-group

##### Response

The response will verify the value under the key “ParameterValue” and “ParameterName”

{

"Parameters": [

<Omitted text>

{

"Description": "parameter for audit logging purpose",

"DataType": "boolean",

"IsModifiable": true,

"AllowedValues": "true,false",

"Source": "engine-default",

"ParameterValue": "true",

"ParameterName": "enable\_user\_activity\_logging",

"ApplyType": "static"

},

<Omitted text>

#### Scenario - Verify Logging to S3 Bucket

You can verify if logging to S3 is enabled by describing the logging status which will return to the Amazon S3 Bucket Name

##### AWS CLI - describe-logging-status

aws redshift describe-logging-status --cluster-identifier mycluster

##### Response

{

"LoggingEnabled": true,

"BucketName": "<S3 Bucket Name>"

}

##### AWS CloudFormation Snippets

In CloudFormation you can create a cluster parameter group with “enable\_user\_activity\_logging” enabled in Parameters property.

myClusterParameterGroup:

Type: "AWS::Redshift::ClusterParameterGroup"

Properties:

Description: "My parameter group"

ParameterGroupFamily: "redshift-1.0"

Parameters:

-

ParameterName: "enable\_user\_activity\_logging"

ParameterValue: "true"

Parameters

A list of parameter names and values that are allowed by the Amazon Redshift engine version that you specified in the ParameterGroupFamily property. For more information, see [Amazon Redshift Parameter Groups](https://docs.aws.amazon.com/redshift/latest/mgmt/working-with-parameter-groups.html) in the *Amazon Redshift Cluster Management Guide*.

*Required*: No

*Type*: [Amazon Redshift Parameter Type](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-property-redshift-clusterparametergroup-parameter.html)

*Update requires*: [No interruption](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-update-behaviors.html#update-no-interrupt)

### Redshift Cluster And Databases Should Not Use Default Parameter Groups

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Ensure that no data or Redshift cluster uses a default parameter group (parameter group = default.redshift-1.0) | I can verify that default parameter groups are not being used |

In Amazon Redshift, you associate a parameter group with each cluster that you create. The parameter group is a group of parameters that apply to all of the databases that you create in the cluster. These parameters configure database settings such as query timeout and date style.

Amazon Redshift provides one default parameter group for each parameter group family. The default parameter group has preset values for each of its parameters, and it cannot be modified. At this time, redshift-1.0 is the only version of the Amazon Redshift engine. Consequently, default.redshift-1.0 is the only default parameter group.

If you want to use different parameter values than the default parameter group, you must create a custom parameter group and then associate your cluster with it. Initially, the parameter values in a custom parameter group are the same as in the default parameter group. The initial source for all of the parameters is engine-default because the values are preset by Amazon Redshift. After you change a parameter value, the source changes to user to indicate that the value has been modified from its default value.

#### Scenario - Verifying the Parameter Group of a Redshift cluster

You can verify the Parameter Group of a redshift cluster by describing the cluster. This will return values such as the Parameter Group associated with the cluster.

##### AWS CLI - describe-clusters

aws redshift describe-clusters --cluster-identifier mycluster

##### Response

Under the “ClusterParameterGroups” key there will be “ParameterGroupName” key that contains the value of the parameter group name associated with the cluster. The value should not be “default.redshift-1.0”, the default parameter group.

{

"Clusters": [

{

<Omitted text>

"ClusterParameterGroups": [

{

"ClusterParameterStatusList": [

{

"ParameterName": "enable\_user\_activity\_logging",

"ParameterApplyStatus": "in-sync"

},

{

"ParameterName": "max\_cursor\_result\_set\_size",

"ParameterApplyStatus": "in-sync"

},

{

"ParameterName": "query\_group",

"ParameterApplyStatus": "in-sync"

},

{

"ParameterName": "datestyle",

"ParameterApplyStatus": "in-sync"

},

{

"ParameterName": "extra\_float\_digits",

"ParameterApplyStatus": "in-sync"

},

{

"ParameterName": "search\_path",

"ParameterApplyStatus": "in-sync"

},

{

"ParameterName": "statement\_timeout",

"ParameterApplyStatus": "in-sync"

},

{

"ParameterName": "wlm\_json\_configuration",

"ParameterApplyStatus": "in-sync"

},

{

"ParameterName": "require\_ssl",

"ParameterApplyStatus": "in-sync"

},

{

"ParameterName": "use\_fips\_ssl",

"ParameterApplyStatus": "in-sync"

}

],

"ParameterGroupName": "test",

"ParameterApplyStatus": "in-sync"

}

],

<Omitted text>

}

]

}

#### Scenario - Create cluster with a non-default parameter group

You can declare the parameter group when creating a cluster.

##### AWS CLI - create-clusters

You can declare the parameter group by using the --cluster-parameter-group-name option.

aws redshift create-cluster --node-type ds2.xlarge --number-of-nodes 2 --master-username adminuser --master-user-pass TopSecret1 --cluster-identifier mycluster --cluster-parameter-group-name test

##### Response

In the response under the “ClusterParameterGroups” key there will be a key “ParameterGroupName” which will contain the value of the parameter group.

{

"Cluster": {

<Omitted text>

"ClusterParameterGroups": [

{

"ParameterGroupName": "test",

"ParameterApplyStatus": "in-sync"

}

],

<Omitted text>

}

}

}

##### Advanced Monitoring Redshift User Only Has Required Access

|  |  |  |
| --- | --- | --- |
| **As A** | **I Want to** | **So that** |
| User | Ensure the Redshift user created for Advanced Monitoring only has required access of “grant select on all tables in schema pg\_catalog to tamreporting” only |  |

Redshift Advance Monitoring is a GitHub project that provides an advance monitoring system for Amazon Redshift that is completely serverless, based on AWS Lambda and Amazon CloudWatch. A serverless Lambda function runs on a schedule, connects to the configured Redshift cluster, and generates CloudWatch custom alarms for common possible issues.

#### Scenario - Create Redshift User with Required Access

##### Redshift Commands - CREATE GROUP, CREATE USER, GRANT SELECT

In this example we first create a group named “TAM\_USERS”. Next we create a user “TAMREPORTING” and place that user in the “TAM\_USERS” group. This is followed by granting select on all tables in the schema “pg\_catalog” to the group “TAM\_USERS”

CREATE GROUP TAM\_USERS;

CREATE USER TAMREPORTING IN GROUP TAM\_USERS WITH PASSWORD '@AbC4321!';

GRANT SELECT ON ALL TABLES in SCHEMA pg\_catalog TO GROUP TAM\_USERS;