aws

Management Guide

Amazon Redshift



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Redshift: Management Guide

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

..... xvii

What Is Amazon Redshift? 1

Are you a first-time Amazon Redshift user? 1

Amazon Redshift Serverless feature overview 2

Amazon Redshift provisioned clusters overview 5

Cluster management 5

Cluster access and security 6

Monitoring clusters 7

Databases 8

Comparing Amazon Redshift Serverless to an Amazon Redshift provisioned data warehouse 9

Using the Amazon Redshift management interfaces for provisioned clusters 34

Working with AWS SDKs 35

Signing an HTTP request 36

Setting up the Amazon Redshift CLI 42

Amazon Redshift Serverless 44

What is Amazon Redshift Serverless? 44

Amazon Redshift Serverless console 45

Considerations when using Amazon Redshift Serverless 48

Compute capacity for Amazon Redshift Serverless 53

Considerations and limitations for serverless endpoint capacity 54

AI-driven scaling and optimization 56

Billing for Amazon Redshift Serverless 62

Billing for compute capacity 62

Billing for on-demand compute capacity 62

Billing for serverless reservations 67

Billing for storage 73

Using the Amazon Redshift Serverless free trial 74

Billing usage notes 74

Amazon Redshift Serverless billing with connection pooling 76

Cost optimization for Amazon Redshift Serverless with zero-ETL 77

Connecting to Amazon Redshift Serverless 77

Connecting to Amazon Redshift Serverless 78

Connecting to Amazon Redshift Serverless through JDBC drivers 78

Connecting to Amazon Redshift Serverless with the Data API ... 80
Connecting with SSL to Amazon Redshift Serverless ... 80
Connecting to Amazon Redshift Serverless from an Amazon Redshift managed VPC endpoint ... 82
Connecting to Amazon Redshift Serverless from an interface VPC endpoint (AWS PrivateLink) ... 83
Connecting to Amazon Redshift Serverless from a Redshift VPC endpoint in another account ... 83
Additional resources ... 88
Defining database roles for federated users ... 88
Defining database roles ... 89
Use cases for defining database roles to grant to federated users ... 89
Additional resources ... 91
Identity and access management in Amazon Redshift Serverless ... 91
Granting permissions ... 92
Getting started with IAM credentials ... 93
Accessing database objects with database-role permissions ... 94
Migrating a provisioned cluster to Amazon Redshift Serverless ... 96
Creating a snapshot of your provisioned cluster ... 96
Connecting to Amazon Redshift Serverless using a driver ... 97
Using the Amazon Redshift Serverless SDK ... 103
Workgroups and namespaces ... 104
Workgroups and namespaces using the console ... 104
Workgroups and namespaces using the AWS Command Line Interface and Amazon Redshift Serverless API ... 105
Workgroups ... 106
Namespaces ... 112
Monitoring queries and workloads ... 115
Adding a query monitoring policy ... 117
Granting query monitoring permissions for a user ... 118
Granting query monitoring permissions for a role ... 119
Setting usage limits ... 119
Setting query limits ... 120
Setting query queues ... 120
Checking summary data using the dashboard ... 125
Audit logging ... 126

Log events in CloudWatch ... 126
CloudWatch metrics ... 127
Snapshots and recovery points ... 136
AWS Backup integration ... 137
Creating a snapshot ... 137
Creating a final snapshot ... 138
Sharing a snapshot or removing snapshot permissions ... 139
Scheduling a snapshot ... 139
Updating a snapshot retention period ... 142
Deleting a snapshot ... 143
Restoring a snapshot ... 143
Converting a recovery point ... 145
Restoring a recovery point ... 145
Copying backups to another AWS Region ... 145
Restoring a table ... 147
Data sharing ... 148
Considerations ... 149
Granting access to view datashares ... 149
Registering namespaces to the AWS Glue Data Catalog ... 150
Tagging resources ... 150
Amazon Redshift provisioned clusters ... 153
Clusters and nodes ... 153
Node type details ... 155
Determining the number of nodes ... 159
Use EC2 to create your cluster ... 159
Amazon Virtual Private Cloud (Amazon VPC) ... 160
Default disk space alarm ... 160
Cluster status ... 161
Considerations for using provisioned clusters ... 163
Region and Availability Zone considerations ... 164
Cluster maintenance ... 164
Cluster operations ... 179
Creating a cluster ... 179
Creating a disk space alarm ... 183
Viewing a cluster ... 183
Modifying a cluster ... 184

Resizing a cluster ... 185
Renaming a cluster ... 201
Upgrading the release version of a cluster ... 202
Pausing and resuming a cluster ... 203
Rebooting a cluster ... 205
Relocating a cluster ... 205
Setting a usage limit ... 209
Shutting down and deleting a cluster ... 211
Snapshots and backups ... 212
Registering a cluster to the AWS Glue Data Catalog ... 237
Multi-AZ deployment ... 238
Setting up Multi-AZ deployment ... 238
Setting up Multi-AZ when creating a new cluster ... 242
Setting up Multi-AZ for a data warehouse restored from a snapshot ... 244
Converting a Single-AZ data warehouse to a Multi-AZ data warehouse ... 245
Converting a Multi-AZ data warehouse to a Single-AZ data warehouse ... 247
Resizing a Multi-AZ data warehouse ... 248
Failing over Multi-AZ deployment ... 249
Viewing queries and loads for Multi-AZ data warehouses ... 251
Monitoring a query in a Multi-AZ deployment ... 252
Terminating a query for a cluster ... 253
Monitoring cluster performance ... 253
Performance data ... 254
Viewing performance data ... 270
Analyzing query execution ... 293
Creating an alarm ... 295
Ending a running query ... 296
Performance metrics in the CloudWatch console ... 296
Query profiler ... 298
Query and Database Monitoring ... 307
Sys View-based Queries and Database Monitoring ... 314
Tracks ... 317
Switching between tracks ... 318
Tracks and restore ... 318
Updating serverless workgroups ... 319
Managing versions ... 320

Determining the workgroup or cluster version ... 320
Zero-ETL integrations ... 322
Considerations ... 324
Considerations when using history mode on the target ... 326
Considerations when the zero-ETL integration source is Aurora or Amazon RDS ... 327
Considerations when the zero-ETL integration source is DynamoDB ... 328
Considerations when the zero-ETL integration source is applications, such as, Salesforce, SAP, ServiceNow, and Zendesk ... 329
Considerations when restoring a zero-ETL integration target ... 330
Considerations when resizing a zero-ETL integration target ... 331
Getting started with zero-ETL integrations ... 331
Create and configure a target Amazon Redshift data warehouse ... 332
Turn on case sensitivity ... 333
Configure authorization in Amazon Redshift ... 335
Create a zero-ETL integration ... 340
Creating destination databases ... 352
Querying replicated data ... 354
Querying replicated data with materialized views ... 355
Querying replicated data from DynamoDB ... 357
Viewing zero-ETL integrations ... 358
History mode ... 361
Sharing your data ... 363
Monitoring zero-ETL integrations ... 364
Monitoring zero-ETL integrations with Amazon Redshift system views ... 364
Monitoring zero-ETL integrations with Amazon EventBridge ... 365
Metrics for zero-ETL integrations ... 365
Modify a zero-ETL integration for DynamoDB ... 370
Delete a zero-ETL integration for DynamoDB ... 371
Supported Regions ... 373
Aurora MySQL ... 373
Aurora PostgreSQL ... 375
Amazon DynamoDB ... 377
Amazon RDS for MySQL ... 379
Enterprise applications ... 380
Troubleshooting zero-ETL integrations ... 382
Query a database ... 398

Connecting to Amazon Redshift ... 398
Querying a database using the Amazon Redshift query editor v2 ... 399
Configuring your AWS account ... 400
Opening query editor v2 ... 407
Connecting to an Amazon Redshift database ... 412
Browsing an Amazon Redshift database ... 414
Creating database objects ... 416
Viewing query and tab history ... 423
Interacting with Amazon Q generative SQL ... 424
Loading data into a database ... 437
Authoring queries ... 446
Notebooks ... 452
Querying the AWS Glue Data Catalog ... 455
Querying a data lake ... 458
Datashares ... 461
Scheduled queries ... 464
Visualizing results ... 474
Collaborating and sharing as a team ... 480
Querying a database using the query editor v1 ... 481
Considerations ... 482
Connecting to a data warehouse using SQL client tools ... 483
Recommendations for connecting with client tools ... 483
Configuring connections in Amazon Redshift ... 484
Best practices for discovering metadata ... 693
Configuring security options for connections ... 701
Connecting from client tools and code ... 709
Using an authentication profile to connect to Amazon Redshift ... 758
Troubleshooting connection issues in Amazon Redshift ... 761
Using the Data API ... 769
Working with the Data API ... 770
Considerations when calling the Data API ... 771
Choosing database authentication credentials ... 773
Mapping JDBC data types ... 774
Running SQL statements with parameters ... 775
Running SQL statements with an idempotency token ... 777
Running SQL statements with session reuse ... 779

Fetching results ... 780
Authorizing access ... 783
Trusted identity propagation ... 792
Calling the Data API ... 795
Troubleshooting Data API issues ... 831
Scheduling Data API operations with Amazon EventBridge ... 832
Monitoring the Data API ... 836
Using AWS KMS with the Data API ... 838
Using Amazon Sagemaker Unified Studio ... 841
Parameter groups ... 843
Default parameter values ... 844
Workload management ... 846
WLM dynamic and static properties ... 846
Properties for the WLM configuration parameter ... 846
Configuring the WLM parameter using the AWS CLI ... 853
Creating a parameter group ... 862
Modifying a parameter group ... 862
Creating a query monitoring rule ... 867
Deleting a parameter group ... 868
Integrate with an AWS Partner ... 869
Loading data with AWS partners ... 870
Reserved nodes ... 872
Reserved node offerings ... 872
Comparing pricing among reserved node offerings ... 873
How reserved nodes work ... 875
Reserved nodes and consolidated billing ... 876
Things to consider before changing node type on a cluster that uses Reserved Instances ... 876
Reserved node examples ... 877
Example 1 ... 877
Example 2 ... 877
Example 3 ... 877
Example 4 ... 878
Example 5 ... 878
Example 6 ... 878
Purchasing a reserved node ... 878
Security ... 881

Data protection ... 883
Data encryption ... 884
Data tokenization ... 902
Routing internetwork traffic ... 903
Identity and access management ... 904
Authentication with identities ... 904
Cross-service confused deputy prevention ... 905
Access control ... 907
Overview of managing access ... 907
Using identity-based policies (IAM policies) ... 914
Native identity provider (IdP) federation ... 976
Using service-linked roles ... 1010
Using IAM authentication to generate database user credentials ... 1018
Authorizing access to AWS services ... 1074
Managing admin passwords ... 1110
Permissions required for AWS Secrets Manager integration ... 1110
Admin password secret rotation ... 1111
Considerations using AWS Secrets Manager with Amazon Redshift ... 1112
Retrieving the ARN of the secret ... 1112
Creating a secret for database connection credentials ... 1113
Logging and monitoring ... 1116
Database audit logging ... 1117
Logging with CloudTrail ... 1133
System table integration with S3 Tables ... 1145
Compliance validation ... 1158
Resilience ... 1160
Infrastructure security ... 1161
Network isolation ... 903
Security groups ... 1162
Interface VPC endpoints ... 1162
Configuration and vulnerability analysis ... 1170
Using Identity Center authentication ... 1171
Prerequisites ... 1171
How Identity Center authentication works ... 1172
GetIdentityCenterAuthToken API operations ... 1173
GetIdentityCenterAuthToken for provisioned clusters ... 1173

GetIdentityCenterAuthToken for serverless workgroups ... 1174
Response syntax ... 1175
Driver integration ... 1175
Direct token usage ... 1176
Java code example ... 1176
IAM policy requirements ... 1177
API permissions ... 1177
Example IAM policies ... 1177
Regional availability ... 1179
Troubleshooting ... 1179
API errors ... 1179
Connection errors ... 1180
Networking tasks ... 1181
Custom domain names for client connections ... 1181
Registering a domain name ... 1182
Requesting a certificate for a domain name ... 1183
Configuring a custom domain ... 1184
Connecting to your provisioned cluster or workgroup ... 1186
Renaming a cluster that has a custom domain assigned ... 1187
Describing custom domain associations ... 1187
Associating a custom domain with a different certificate ... 1188
Deleting a custom domain ... 1189
Redshift-managed VPC endpoints ... 1190
Considerations ... 1191
Granting access to a VPC ... 1192
Creating a Redshift-managed VPC endpoint ... 1192
Redshift resources in a VPC ... 1193
Creating a cluster or workgroup in a VPC ... 1196
VPC security groups ... 1197
Configuring security settings for a cluster or workgroup ... 1199
Subnets for Redshift resources ... 1202
Blocking public access to VPCs and subnets ... 1205
Controlling network traffic with enhanced VPC routing ... 1206
Controlling database traffic with VPC endpoints ... 1208
Turning on enhanced VPC routing ... 1209
Querying data lake tables with enhanced VPC routing ... 1210

Events 1216
Cluster event notification subscriptions 1216
Creating an event notification subscription 1219
Provisioned cluster event notifications 1219
Amazon Redshift Serverless event notifications 1241
Zero-ETL integration event notifications 1254
Quotas and limits 1265
Quotas for Amazon Redshift objects 1265
Quotas for Amazon Redshift Serverless objects 1274
Quotas for Amazon Redshift Data API 1277
Quotas for query editor v2 objects 1280
Quotas and limits for Amazon Redshift Spectrum objects 1281
Naming constraints 1282
Tag resources 1286
Tagging requirements 1287
Managing resource tags 1287
AWS Backup integration 1289
Considerations 1290
Limitations 1291
Using AWS Backup with Amazon Redshift 1291
Cluster versions 1292
Patch 206 1294
New features 1294
Patch 205 1296
New features 1296
Patch 204 1298
New features 1298
Patch 203 1302
New features 1302
Patch 202 1305
New features 1306
Patch 201 1308
New features 1308
Patch 200 1310
New features 1310
Patch 199 1313

New features 1313
Patch 198 1315
New features 1316
Patch 197 1317
New features 1317
Patch 196 1319
New features 1320
Patch 195 1320
New features 1320
Patch 194 1321
New features 1321
Patch 193 1322
New features 1322
Patch 192 1323
New features 1323
Patch 191 1323
New features 1324
Patch 190 1324
New features 1324
Patch 189 1325
New features 1325
Patch 188 1326
New features 1326
Patch 187 1327
New features 1328
Patch 186 1329
New features 1330
Patch 185 1330
New features 1331
Patch 184 1332
New features 1333
Patch 183 1334
New features 1334
Patch 182 1335
New features 1336
Patch 181 1337

New features 1338
Patch 180 1339
New features 1340
Patch 179 1341
New features 1341
Patch 178 1342
New features 1343
Patch 177 1345
New features 1346
Patch 176 1347
New features 1348
Patch 175 1349
New features 1349
Patch 174 1350
New features for this version 1350
New features for this version 1350
New features for this version 1350
New features for this version 1350
New features for this version 1350
New features for this version 1350
New features for this version 1350
Patch 173 1351
New features for this version 1351
New features for this version 1351
New features for this version 1351
New features for this version 1351
New features for this version 1351
New features for this version 1351
New features for this version 1351
New features for this version 1351
New features for this version 1351
New features for this version 1351
Patch 172 1353
New features 1353
Patch 171 1354
New features 1354

Patch 170 1354
New features 1354
Patch 169 1354
New features 1355
Patch 168 1355
New features 1355
Behavior changes 1357
Upcoming behavior changes 1357
SUPER data type supports larger individual strings starting with Patch 206 1358
AWS KMS key permission enforcement for Amazon Redshift Serverless APIs after August 17, 2026 1359
Amazon Redshift enforces user lockout after multiple failed login attempts starting with Patch 208 1360
Enhanced billing model for manual snapshots on Amazon Redshift Serverless and Amazon Redshift RG instances effective June 08, 2026 1361
Iceberg DELETE on Lake Formation tables requires DELETE permission starting with Patch 202 1361
Amazon Redshift Serverless preserves zero-ETL and S3 event integrations on snapshot restore starting with Patch 202 1362
End of support for the Amazon Redshift ODBC 1.x driver on December 31, 2026 1362
Scalar Python UDFs will reach end of support after June 30, 2026 1363
Materialized View (MV) Auto-REFRESH Behavior Change after February 27, 2026 1364
Amazon Redshift won't support functions that access consumer information through datasharing after February 16, 2026 1364
Minimum Transport Layer Security (TLS) version changes effective starting October 31, 2026 1364
Amazon Redshift won't support the creation of new scalar Python UDFs after October 30, 2025 1365
Recent behavior changes 1366
Amazon Redshift takes automated snapshots of paused data sharing producer clusters starting September 30, 2026 1366
Amazon Redshift uses up-to-date IANA Time Zone Database after Aug 26, 2025 1366
Amazon Redshift Serverless RPU changes effective after August 15, 2025 1367
Database audit logging changes effective after August 10, 2025 1368
Virtual Private Cloud Endpoint changes for serverless workgroups effective after June 27, 2025 1368

Query monitoring changes effective after May 2, 2025 1368
Security changes effective after January 10, 2025 1369
Code examples 1372
Basics 1373
Hello Amazon Redshift 1373
Learn the basics 1378
Actions 1444
Scenarios 1509
Create a web application to track Amazon Redshift data 1509
Get started with serverless data warehouses 1510
Getting started with provisioned data warehouse clusters 1527
Document history 1539

Amazon Redshift will no longer support the use of Python UDFs after June 30, 2026. We will start enforcing it in phases. For more information on the details of Python end of life and migration options, see the blog post that was published on June 30, 2025.

What is Amazon Redshift?

Welcome to the *Amazon Redshift Management Guide*. Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. Amazon Redshift Serverless lets you access and analyze data without all of the configurations of a provisioned data warehouse. Resources are automatically provisioned and data warehouse capacity is intelligently scaled to deliver fast performance for even the most demanding and unpredictable workloads. You don't incur charges when the data warehouse is idle, so you only pay for what you use. You can load data and start querying right away in the Amazon Redshift query editor v2 or in your favorite business intelligence (BI) tool. Enjoy the best price performance and familiar SQL features in an easy-to-use, zero administration environment.

Regardless of the size of the dataset, Amazon Redshift offers fast query performance using the same SQL-based tools and business intelligence applications that you use today.

Are you a first-time Amazon Redshift user?

If you are a first-time user of Amazon Redshift, we recommend that you begin by reading the following sections:

- Service Highlights and Pricing – This product detail page provides the Amazon Redshift value proposition, service highlights, and pricing.
- Get started with Amazon Redshift Serverless data warehouses – This topic walks you through the process of setting up a serverless data warehouse, creating resources, and querying sample data.
- Amazon Redshift Database Developer Guide – If you are a database developer, this guide explains how to design, build, query, and maintain the databases that make up your data warehouse.

If you prefer to manage your Amazon Redshift resources manually, you can create provisioned clusters for your data querying needs. For more information, see Amazon Redshift clusters.

As an application developer, you can use the Amazon Redshift API or the AWS Software Development Kit (SDK) libraries to manage clusters programmatically. If you use the Amazon Redshift API, you must authenticate every HTTP or HTTPS request to the API by signing it. For more information about signing requests, go to Signing an HTTP request.

For information about the API, CLI, and SDKs, go to the following links:

- Amazon Redshift Serverless API Reference
- Amazon Redshift API Reference
- Amazon Redshift Data API API Reference
- AWS CLI Command Reference
- SDK References in Tools for Amazon Web Services.

Amazon Redshift Serverless feature overview

Most of the features supported by an Amazon Redshift provisioned data warehouse are also supported by Amazon Redshift Serverless. The following are some of its key capabilities.

Feature	Description
Snapshots	You can restore a snapshot of Amazon Redshift Serverless or a provisioned data warehouse to Amazon Redshift Serverless. For more information, see Snapshots and recovery points.
Recovery points	Amazon Redshift Serverless automatically creates a point of recovery every 30 minutes. These recovery points are kept for 24 hours. You can use them to restore after accidental writes or deletes. When you restore from a recovery point, all the data in your Amazon Redshift Serverless database is restored to an earlier point in time. You can also create a snapshot from a recovery point if you need to keep a point of recovery for a longer period. For more information, see Snapshots and recovery points.
Base RPU capacity	You can set a base capacity in Redshift Processing Units (RPUs). One RPU provides 16 GB of memory. This setting gives you the ability to control the balance between resources in use and cost for your workload. You can increase this value to grow resources available and improve query performance, or lower the value to limit your spending. The default is 128 RPUs. You can also set usage limits, such as RPUs used per day, to control costs. For more information, see Billing for Amazon Redshift Serverless.
Usage limits of data sharing	You can limit the amount of data transferred from a producer Region to a consumer Region using the console or the API. These data transfer costs differ by AWS Region, and are measured in terabytes. For more information about

Feature	Description
	data sharing, see Getting started data sharing using the console in the *Amazon Redshift Database Developer Guide*.
User-defined functions (UDFs)	You can run user-defined functions (UDFs) in Amazon Redshift Serverless. For more information, see Creating user-defined functions in the *Amazon Redshift Database Developer Guide*.
Stored procedures	You can run stored procedures in Amazon Redshift Serverless. For more information, see Creating stored procedures in the *Amazon Redshift Database Developer Guide*.
Materialized views	You can create materialized views in Amazon Redshift Serverless. For more information, see Creating materialized views in the *Amazon Redshift Database Developer Guide*.
Spatial functions	You can run spatial functions in Amazon Redshift Serverless. For more informati on, see Querying spatial data in the *Amazon Redshift Database Developer Guide*.
Federated queries	You can run queries to join data with Aurora DB cluster and Amazon RDS databases from Amazon Redshift Serverless. For more information, see Querying data with federated queries in the *Amazon Redshift Database Developer Guide*.
Data lake queries	You can run queries to join data from your Amazon S3 data lake with Amazon Redshift Serverless. For more information, see Querying a data lake in the *Amazon Redshift Management Guide*.
HyperLogL og	You can run HyperLogLog functions in Amazon Redshift Serverless. For more information, see Using HyperLogLog sketches in the *Amazon Redshift Database Developer Guide*.
Querying data across databases	You can query data across databases with Amazon Redshift Serverless. For more information, see Querying data across databases in the *Amazon Redshift Database Developer Guide*.

Feature	Description
Data sharing	You can access datashares on provisioned data warehouses with Amazon Redshift Serverless. For more information, see Sharing data across clusters in the *Amazon Redshift Database Developer Guide*.
Semi-structured data querying	You can ingest and store semi-structured data with the SUPER data type with Amazon Redshift Serverless. For more information, see Ingesting and querying semi-structured data in the *Amazon Redshift Database Developer Guide*.
Tagging resources	You can use the AWS CLI or the Amazon Redshift Serverless API to tag resources with metadata related to the resource. For more information, see Tagging resources.
Machine learning	You can use Amazon Redshift machine learning with Amazon Redshift Serverless. For more information, see Using machine learning in the *Amazon Redshift Database Developer Guide*.
SQL commands and functions	With a few exceptions (such as REBOOT_CLUSTER), you can use Amazon Redshift SQL commands and functions with Amazon Redshift Serverless. For more information, see SQL reference in the *Amazon Redshift Database Developer Guide*.
CloudFormation resources	Using CloudFormation templates, you can deploy and update Amazon Redshift Serverless resources. This integration means you can spend less time managing resources and focus on your applications. For more information about CloudFormation resources in Amazon Redshift Serverless, see Amazon Redshift Serverless resource type reference.
CloudTrail resources	Amazon Redshift Serverless is integrated with AWS CloudTrail to provide a record of actions taken in Amazon Redshift Serverless. CloudTrail captures all API calls for Amazon Redshift Serverless as events. For more information, see CloudTrail for Amazon Redshift Serverless.

Amazon Redshift provisioned clusters overview

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.

The following video shows you how to create a cluster and query data using the Amazon Redshift query editor v2.

Cluster management

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 run, and the query runtime performance that you need.

Creating and managing clusters

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 without any interruption to the service. For more information, see Amazon Redshift provisioned clusters.

Reserving compute nodes

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. For more information, see Reserved nodes.

Creating cluster snapshots

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. For more information about snapshots, see Amazon Redshift snapshots and backups.

Cluster access and security

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. For more information about database security, see Managing Database Security in the *Amazon Redshift Database Developer Guide*.

AWS accounts and IAM credentials

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. For more information, see Security in Amazon Redshift. For more information about managing IAM identities, including guidance and best practices for IAM roles, see Identity and access management in Amazon Redshift.

Security groups

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-VPC platform for your Amazon Redshift cluster, you must use VPC security groups. We recommend that you launch your cluster in an EC2-VPC platform.

  You cannot move a cluster to a VPC after it has been launched with EC2-Classic. However, you can restore an EC2-Classic snapshot to an EC2-VPC cluster using the Amazon Redshift console. For more information, see Restoring a cluster from a snapshot.
- If you are using the EC2-Classic platform for your Amazon Redshift cluster, you must use Amazon Redshift 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. For more information, see Amazon Redshift security groups.

In addition to the inbound access rules, you create database users to provide credentials to authenticate to the database within the cluster itself. For more information, see Databases in this topic.

Encryption

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 cluster that is not encrypted 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.

For more information about encryption, keys, and hardware security modules, see Amazon Redshift database encryption.

SSL connections

You can use Secure Sockets Layer (SSL) encryption to encrypt the connection between your SQL client and your cluster. For more information, see Configuring security options for connections.

Monitoring clusters

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. Use the metrics in Amazon Redshift and Amazon CloudWatch to learn about the health and performance of your clusters and databases.

Database audit logging

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. For more information, see Database audit logging.

Events and notifications

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 inform you that the event has occurred. For more information about events and notifications, see Amazon Redshift events.

Performance

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. For more information about performance metrics and monitoring, see Monitoring Amazon Redshift cluster performance.

Databases

Amazon Redshift creates one database when you provision a cluster. This is the database that you use to load data and run queries on your data. You can create additional databases as needed by running a SQL command. For more information about creating additional databases, go to Step 1: Create a database in the *Amazon Redshift Database Developer Guide*.

When you provision a cluster, you specify an admin user who has access to all of the databases that are created within the cluster. This admin user is a superuser who is the only user with access to the database initially, though this user can create additional superusers and users. For more information, go to Superusers and Users in the *Amazon Redshift Database Developer Guide*.

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. For more information, see Amazon Redshift parameter groups.

For more information about databases in Amazon Redshift, go to the Amazon Redshift Database Developer Guide.

Comparing Amazon Redshift Serverless to an Amazon Redshift provisioned data warehouse

For Amazon Redshift Serverless, some concepts and features are different than their corresponding feature for an Amazon Redshift provisioned data warehouse. For instance, one contrasting comparison is that Amazon Redshift Serverless doesn't have the concept of a cluster or node. The following table describes features and behavior in Amazon Redshift Serverless and explains how they differ from the equivalent feature in a provisioned data warehouse.

Feature	Description	Serverless	Provisioned
Workgroup and Namespace	To isolate workloads and manage different resources in Amazon Redshift Serverless, you can create namespaces and workgroups in order to manage storage and compute resources separately.	A namespace is a collection of database objects and users. A workgroup is a collection of compute resources. For more information, see Amazon Redshift Serverless to understand the	A provisioned cluster is a collection of compute nodes and a leader node, which you manage directly. For more information, see Amazon Redshift provisioned clusters.

Feature	Description	Serverless	Provisioned
		design for Amazon Redshift Serverless.	

Feature	Description	Serverless	Provisioned
Node types	When you work with Amazon Redshift Serverless, you don't choose node types or specify node count like you do with a provisioned Amazon Redshift cluster.	Amazon Redshift Serverless automatically provisions and manages capacity for you. You can optionally specify base data warehouse capacity to select the right price/performance balance for your workloads. You can also specify maximum RPU hours to set cost controls to make sure that costs are	You build a cluster with node types that meet your cost and performance specifications. For more information, see Amazon Redshift provisioned clusters.

Feature	Description	Serverless	Provisioned
		predictable. For more information, see Compute capacity for Amazon Redshift Serverless.	
Workload management and concurrency scaling	Amazon Redshift can scale for periods of heavy load. Amazon Redshift Serverless also can scale to meet intermittent periods of high load.	Amazon Redshift Serverless automatically manages resources efficiently and scales, based on workloads, within the thresholds of cost controls. For more information, see Billing for compute capacity.	With a provisioned data warehouse, you enable concurrency scaling on your cluster to handle periods of heavy load. For more information, see Concurrency scaling.

Feature	Description	Serverless	Provisioned
Port	The port number that you use to connect.	With Amazon Redshift Serverless, you can change to another port from the port range of 5431–5455 or 8191–8215. For more information, see Connecting to Amazon Redshift Serverless.	With a provisioned data warehouse, you can choose any port to connect.

Feature	Description	Serverless	Provisioned
Resizing	Add or remove compute resources to perform well for the workload.	Resizing is not applicable in Amazon Redshift Serverless. You can however change the base data warehouse RPU capacity, based on your price and performance requirements. For more information, see Compute capacity for Amazon Redshift Serverless.	With a provisioned cluster, you perform a cluster resize to add nodes or remove nodes. For more information, see Overview of managing clusters in Amazon Redshift.

Feature	Description	Serverless	Provisioned
Pausing and resuming	You can pause a provisioned cluster when you don't have workloads to run, to save cost.	With Amazon Redshift Serverless, you pay only when queries run, so there is no need to pause or resume. For more information, see Billing for compute capacity.	You pause and resume a cluster manually, based on an assessment of your workload at various times. For more information, see Overview of managing clusters in Amazon Redshift.

Feature	Description	Serverless	Provisioned
Querying external data	You can query data in Amazon S3 buckets, in a variety of formats, such as JSON, Parquet or Iceberg.	Amazon Redshift Serverless includes an integrated data lake query engine that runs on the workgroup's compute resources, and supports querying Amazon S3 data within an AWS Region or between different AWS Regions. To tune data lake query performance, adjust the base RPU capacity of your	With RA3 and DC2 provisioned clusters, data lake queries run on Redshift Spectrum using dedicated servers independent of your cluster. Redshift Spectrum requires the Amazon S3 bucket to be in the same AWS Region as the cluster. For more information, see Querying external data using Amazon Redshift Spectrum. RG provisioned clusters include an integrated data lake query engine. Data lake queries run directly on the cluster's own compute resources, similar to Redshift Serverless, and support querying Amazon S3 data in the same AWS Region or in a different AWS Region. For more information, see Querying your data lake.

Feature	Description	Serverless	Provisioned
		workgroup.	
Compute-resource billing	How billing accrues for Amazon Redshift vs Amazon Redshift Serverless.	With Amazon Redshift Serverless, you pay for the workloads you run, in RPU-hours on a per-second basis, with a 60-second minimum charge. This includes queries that access data in open file formats in Amazon S3. For more information, see Billing for compute capacity.	With a provisioned cluster, billing occurs per second when the cluster isn't paused. For RG clusters, data lake queries are included in the node-hour compute pricing with no separate Spectrum charge. For RA3 and DC2 clusters, data lake queries incur additional Redshift Spectrum charges.

Feature	Description	Serverless	Provisioned
Maintenance window	How server maintenance works.	With Amazon Redshift Serverless, there is no maintenance window. Updates are handled seamlessly. For more information, see What is Amazon Redshift Serverless?	With a provisioned cluster, you specify a maintenance window when patching occurs. (Typically, you choose a recurring time when use is low.)
Encryption	You can enable database encryption.	Amazon Redshift Serverless is always encrypted with AWS KMS, with AWS managed or customer managed keys.	The data in a provisioned data warehouse can be encrypted with AWS KMS (with AWS managed or customer managed keys), or unencrypted. See Amazon Redshift database encryption.

Feature	Description	Serverless	Provisioned
Storage billing	How billing for storage works.	For Amazon Redshift Serverless. The rate is calculated according to GB per month. See Billing for compute capacity.	Storage is billed apart from compute resources for a provisioned cluster with RG or RA3 nodes.

Feature	Description	Serverless	Provisioned
User management	How users are managed.	For Amazon Redshift Serverless, users are IAM or Redshift users. For more information, see Identity and access management in Amazon Redshift Serverless. For more information about managing IAM identities, including best practices for IAM roles, see Identity and access management in	For a provisioned data warehouse, users are IAM or Redshift users. For more information, see Managing database security in the *Amazon Redshift Database Developer Guide*. For more information about managing IAM identities, including best practices for IAM roles, see Identity and access management in Amazon Redshift.

Feature	Description	Serverless	Provisioned
		Amazon Redshift.	

Feature	Description	Serverless	Provisioned
JDBC and ODBC tools and compatibility	How client connections work.	Amazon Redshift Serverless is compatible with any JDBC or ODBC compliant tool or client application. For more information about drivers, see Configuring connections in the *Amazon Redshift Management Guide*. For information about connecting to Amazon Redshift Serverless, see	Amazon Redshift provisioned is compatible with any JDBC or ODBC compliant tool or client application. For more information about drivers, see Configuring connections in the *Amazon Redshift Management Guide*. For information about connecting to clusters, see Connecting to an Amazon Redshift data warehouse using SQL client tools.

Feature	Description	Serverless	Provisioned
		Connecting to Redshift Serverless.	
Requirement for credentials on sign in	How credentials are handled.	For Amazon Redshift Serverless, you don't have to enter credentials in every instance. For more information, see Connecting to Amazon Redshift Serverless.	Access to Amazon Redshift requires sign-in credentials from a user associated with an IAM role. The IAM role has specific permissions attached for a provisioned data warehouse. Once authenticated, the user can connect directly to the database, to the Redshift console, and to query editor v2.

Feature	Description	Serverless	Provisioned
Data API	You can access data from web services and other applications.	Amazon Redshift Serverless supports the Amazon Redshift Data API. With Amazon Redshift Serverless, you use the `workgroup-name` parameter instead of the `cluster-identity` parameter. For more information about calling the Data API, see Using the Amazon Redshift Data API.	Amazon Redshift provisioned supports the Amazon Redshift Data API. With Amazon Redshift clusters, you use the`cluster-identity` parameter instead of the `workgroup-name` parameter. For more information about calling the Data API, see Using the Amazon Redshift Data API.

Feature	Description	Serverless	Provisioned
Snapshots	Provides point-in-time recovery.	Amazon Redshift Serverless supports snapshots and recovery points. For more information about snapshots and recovery points for a namespace, see Snapshots and recovery points.	Provisioned clusters support snapshots. For more information, see Managing snapshots using the console.

Feature	Description	Serverless	Provisioned
Data Sharing	Provides the ability to share data between databases in the same account or in different accounts.	Amazon Redshift Serverless supports all of the data sharing features that a provisioned data warehouse does. It also supports data sharing between Amazon Redshift Serverless and a provisioned data warehouse, tool, or client application.	Provisioned clusters support cross database, cross account, cross-Region, and AWS Data Exchange data sharing. For more information, see Sharing data across clusters in Amazon Redshift.

Feature	Description	Serverless	Provisioned
Tracks	Provides a schedule for software updates.	Amazon Redshift Serverless has no concept of a track. Versions and updates are handled by the service. For more information about the design of Amazon Redshift Serverless, see Snapshots and recovery points.	Provisioned clusters support switching between current and trailing tracks.

Feature	Description	Serverless	Provisioned
System tables and views	Provides a way to monitor your resources and system metadata.	Amazon Redshift Serverless supports new system tables and views. For more information about system tables, see Monitoring queries and workloads with Amazon Redshift Serverless. For information about how to migrate your queries from using the older provisioned system tables and	A provisioned data warehouse supports the existing set of system tables and views for monitoring and other tasks that require system metadata.

Feature	Description	Serverless	Provisioned
		views to the new views, see Migrating to SYS monitoring views.	
Parameter groups	This is a group of parameters that apply to all of the databases created in a cluster. These parameters configure database settings such as query timeout and date style.	Amazon Redshift Serverless does not have the concept of a parameter group.	Provisioned data warehouses support parameter groups. For more information about parameter groups for a provisioned cluster, see Amazon Redshift parameter groups.

Feature	Description	Serverless	Provisioned
Query monitoring	Provides a time-based view of queries run.	Query monitoring in Amazon Redshift Serverless requires users to connect to the database to use system tables. Thus, query monitoring and system tables are in sync. Queries of system tables in Amazon Redshift Serverless use the database user mapped to the IAM user for using	Query monitoring in provisioned clusters does not show all data in system tables.

Feature	Description	Serverless	Provisioned
		query monitoring. For more information about monitoring queries, see Monitoring queries and workloads with Amazon Redshift Serverless.	

Feature	Description	Serverless	Provisioned
Audit logging	Provides information about connections and user activities in the database.	With Amazon Redshift Serverless, CloudWatch is a destination for audit logs. Amazon S3 based audit log delivery is not supported for Amazon Redshift Serverless. For more information, see Audit logging for Amazon Redshift Serverless.	For a provisioned cluster, Amazon S3-based audit log delivery has been the norm. Now, delivery of audit logs to CloudWatch is extended to cover provisioned data warehouses.

Feature	Description	Serverless	Provisioned
Event notifications	Amazon EventBridge is a serverless event bus service that you can use to connect your applications with event data from a variety of sources.	Amazon Redshift Serverless uses Amazon EventBridge to manage event notifications to keep you up-to-date regarding changes in your data warehouse. For more information, see Amazon Redshift Serverless event notifications with Amazon EventBridge.	For a provisioned cluster, you manage event notifications using the Amazon Redshift console to create event subscriptions. For more information, see Creating an event notification subscription.

Feature	Description	Serverless	Provisioned
Cursor constraints	Amazon Redshift enforces constraints on the size of all cursor result sets.	Amazon Redshift Serverless has a cursor maximum total result set size of 150,000 MB.	For a provisioned cluster, the cursor maximum total result set size depends on the cluster type. For more information, see Cursor constraints.

Using the Amazon Redshift management interfaces for provisioned clusters

> **Note**
>
> This topic focuses on Amazon Redshift management interfaces for provisioned clusters. There are similar management interfaces for Amazon Redshift Serverless and Amazon Redshift Data API.

Amazon Redshift supports several management interfaces that you can use to create, manage, and delete Amazon Redshift clusters: the AWS SDKs, the AWS Command Line Interface (AWS CLI), and the Amazon Redshift management API.

The Amazon Redshift API – You can call this Amazon Redshift management API by submitting a request. Requests are HTTP or HTTPS requests that use the HTTP verbs `GET` or `POST` with a parameter named `Action`. Calling the Amazon Redshift API is the most direct way to access the Amazon Redshift service. However, it requires that your application handle low-level details such as error handling and generating a hash to sign the request.

- For information about building and signing an Amazon Redshift API request, see Signing an HTTP request.

- For information about the Amazon Redshift API actions and data types for Amazon Redshift, see the Amazon Redshift API reference.

AWS SDKs – You can use the AWS SDKs to perform Amazon Redshift cluster-related operations. Several of the SDK libraries wrap the underlying Amazon Redshift API. They integrate the API functionality into the specific programming language and handle many of the low-level details, such as calculating signatures, handling request retries, and error handling. Calling the wrapper functions in the SDK libraries can greatly simplify the process of writing an application to manage an Amazon Redshift cluster.

- Amazon Redshift is supported by the AWS SDKs for Java, .NET, PHP, Python, Ruby, and Node.js. The wrapper functions for Amazon Redshift are documented in the reference manual for each SDK. For a list of the AWS SDKs and links to their documentation, see Tools for Amazon Web Services.
- This guide provides examples of working with Amazon Redshift using the Java SDK. For more general AWS SDK code examples, see Code examples for Amazon Redshift using AWS SDKs.

AWS CLI – The CLI provides a set of command line tools that you can use to manage AWS services from Windows, Mac, and Linux computers. The AWS CLI includes commands based on the Amazon Redshift API actions.

- For information about installing and setting up the Amazon Redshift CLI, see Setting up the Amazon Redshift CLI.
- For reference material on the Amazon Redshift CLI commands, see Amazon Redshift in the *AWS CLI Reference*.

Using this service with an AWS SDK

AWS software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that make it easier for developers to build applications in their preferred language.

SDK documentation	Code examples
AWS SDK for C++	AWS SDK for C++ code examples

SDK documentation	Code examples
AWS CLI	AWS CLI code examples
AWS SDK for Go	AWS SDK for Go code examples
AWS SDK for Java	AWS SDK for Java code examples
AWS SDK for JavaScript	AWS SDK for JavaScript code examples
AWS SDK for Kotlin	AWS SDK for Kotlin code examples
AWS SDK for .NET	AWS SDK for .NET code examples
AWS SDK for PHP	AWS SDK for PHP code examples
AWS Tools for PowerShell	AWS Tools for PowerShell code examples
AWS SDK for Python v4	AWS SDK for Python v4 code examples
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) code examples
AWS SDK for Ruby	AWS SDK for Ruby code examples
AWS SDK for Rust	AWS SDK for Rust code examples
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP code examples
AWS SDK for Swift	AWS SDK for Swift code examples

Example availability

Can't find what you need? Request a code example by using the **Provide feedback** link at the bottom of this page.

Signing an HTTP request

Amazon Redshift requires that every request you send to the management API be authenticated with a signature. This topic explains how to sign your requests.

If you are using one of the AWS Software Development Kits (SDKs) or the AWS Command Line Interface, request signing is handled automatically, and you can skip this section. For more information about using AWS SDKs, see Using the Amazon Redshift management interfaces for provisioned clusters. For more information about using the Amazon Redshift Command Line Interface, go to Amazon Redshift command line reference.

To sign a request, you calculate a digital signature by using a cryptographic hash function. A cryptographic hash is a function that returns a unique hash value that is based on the input. The input to the hash function includes the text of your request and your secret access key that you can get from temporary credentials. The hash function returns a hash value that you include in the request as your signature. The signature is part of the `Authorization` header of your request.

Note

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.
To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
IAM	(Recommended) Use console credentials as temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Login for AWS local development in the *AWS Command Line Interface User Guide*. • For AWS SDKs, see Login for AWS local development in the *AWS SDKs and Tools Reference Guide*.

Which user needs programmatic access?	To	By
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.

Which user needs programmatic access?	To	By
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

After Amazon Redshift receives your request, it recalculates the signature by using the same hash function and input that you used to sign the request. If the resulting signature matches the signature in the request, Amazon Redshift processes the request; otherwise, the request is rejected.

Amazon Redshift supports authentication using AWS signature version 4. The process for calculating a signature is composed of three tasks. These tasks are illustrated in the example that follows.

- Task 1: Create a canonical request

  Rearrange your HTTP request into a canonical form. Using a canonical form is necessary because Amazon Redshift uses the same canonical form to calculate the signature it compares with the one you sent.
- Task 2: Create a string to sign

Create a string that you will use as one of the input values to your cryptographic hash function. The string, called the *string to sign*, is a concatenation of the name of the hash algorithm, the request date, a *credential scope* string, and the canonicalized request from the previous task. The *credential scope* string itself is a concatenation of date, region, and service information.

- Task 3: Calculate a signature

  Calculate a signature for your request by using a cryptographic hash function that accepts two input strings, your string to sign and a *derived key*. The derived key is calculated by starting with your secret access key and using the credential scope string to create a series of hash-based message authentication codes (HMAC-SHA256).

Example signature calculation

The following example walks you through the details of creating a signature for CreateCluster request. You can use this example as a reference to check your own signature calculation method. Other reference calculations are included in the Request signature examples section of the IAM User Guide.

You can use a GET or POST request to send requests to Amazon Redshift. The difference between the two is that for the GET request your parameters are sent as query string parameters. For the POST request they are included in the body of the request. The example below shows a POST request.

The example assumes the following:

- The time stamp of the request is `Fri, 07 Dec 2012 00:00:00 GMT`.
- The endpoint is US East (Northern Virginia) Region, `us-east-1`.

The general request syntax is:

```
https://redshift.us-east-1.amazonaws.com/
   ?Action=CreateCluster
   &ClusterIdentifier=examplecluster
   &MasterUsername=masteruser
   &MasterUserPassword=12345678Aa
   &NumberOfNode=2
   &NodeType=dc2.large
   &Version=2012-12-01
```

```
   &x-amz-algorithm=AWS4-HMAC-SHA256
   &x-amz-credential=AKIAIOSFODNN7EXAMPLE/20121207/us-east-1/redshift/aws4_request
   &x-amz-date=20121207T000000Z
   &x-amz-signedheaders=content-type;host;x-amz-date
```

The canonical form of the request calculated for Task 1: Create a Canonical Request is:

```
POST
/

content-type:application/x-www-form-urlencoded; charset=utf-8
host:redshift.us-east-1.amazonaws.com
x-amz-date:20121207T000000Z

content-type;host;x-amz-date
55141b5d2aff6042ccd9d2af808fdf95ac78255e25b823d2dbd720226de1625d
```

The last line of the canonical request is the hash of the request body. The third line in the canonical request is empty because there are no query parameters for this API.

The string to sign for Task 2: Create a String to Sign is:

```
AWS4-HMAC-SHA256
20121207T000000Z
20121207/us-east-1/redshift/aws4_request
06b6bef4f4f060a5558b60c627cc6c5b5b5a959b9902b5ac2187be80cbac0714
```

The first line of the *string to sign* is the algorithm, the second line is the time stamp, the third line is the *credential scope*, and the last line is a hash of the canonical request from Task 1: Create a Canonical Request. The service name to use in the credential scope is `redshift`.

For Task 3: Calculate a Signature, the derived key can be represented as:

```
derived key = HMAC(HMAC(HMAC(HMAC("AWS4" + YourSecretAccessKey,"20121207"),"us-
east-1"),"redshift"),"aws4_request")
```

The derived key is calculated as series of hash functions. Starting from the inner HMAC statement in the formula above, you concatenate the phrase **AWS4** with your secret access key and use this as the key to hash the data "us-east-1". The result of this hash becomes the key for the next hash function.

After you calculate the derived key, you use it in a hash function that accepts two input strings, your string to sign and the derived key. For example, if you use the secret access key `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY` and the string to sign given earlier, then the calculated signature is as follows:

```
9a6b557aa9f38dea83d9215d8f0eae54100877f3e0735d38498d7ae489117920
```

The final step is to construct the `Authorization` header. For the demonstration access key `AKIAIOSFODNN7EXAMPLE`, the header (with line breaks added for readability) is:

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20121207/us-east-1/
redshift/aws4_request,
SignedHeaders=content-type;host;x-amz-date,
Signature=9a6b557aa9f38dea83d9215d8f0eae54100877f3e0735d38498d7ae489117920
```

Setting up the Amazon Redshift CLI

This section explains how to set up and run the AWS CLI command line tools for use in managing Amazon Redshift. The Amazon Redshift command line tools run on the AWS Command Line Interface (AWS CLI), which in turn uses Python (https://www.python.org/). The AWS CLI can be run on any operating system that supports Python.

Installing the AWS Command Line Interface

To begin using the Amazon Redshift command line tools, you first set up the AWS CLI, and then you add configuration files that define the Amazon Redshift CLI options.

If you have already installed and configured the AWS CLI for another AWS service, you can skip this procedure.

To install the AWS Command Line Interface

1. Go to Install or update to the latest version of the AWS CLI, and then follow the instructions for installing the AWS CLI.

   For CLI access, you need an access key ID and a secret access key. Use temporary credentials instead of long-term access keys when possible. Temporary credentials include an access key ID, a secret access key, and a security token that indicates when the credentials expire. For more information, see Using temporary credentials with AWS resources in the *IAM User Guide*.

2. Create a file containing configuration information such as your access keys, default region, and command output format. Then set the `AWS_CONFIG_FILE` environment variable to reference that file. For detailed instructions, go to Configuring the AWS command line interface in the AWS Command Line Interface User Guide.
3. Run a test command to confirm that the AWS CLI interface is working. For example, the following command should display help information for the AWS CLI:

   ```
   aws help
   ```

   The following command should display help information for Amazon Redshift:

   ```
   aws redshift help
   ```

For reference material on the Amazon Redshift CLI commands, go to Amazon Redshift in the AWS CLI Reference.

Amazon Redshift Serverless

Amazon Redshift Serverless makes it convenient for you to run and scale analytics without having to provision and manage an on-premises data warehouse. With Amazon Redshift Serverless, data analysts, developers, and data scientists can now use Amazon Redshift to get insights from data in seconds by loading data into and querying records from the data warehouse in the cloud. Amazon Redshift automatically provisions and scales data warehouse capacity to deliver fast performance for demanding and unpredictable workloads. You pay only for the capacity that you use. You can benefit from this simplicity without changing your existing analytics and business intelligence applications.

For information about Amazon Redshift Serverless SLAs, see Amazon Redshift Service Level Agreement.

What is Amazon Redshift Serverless?

Amazon Redshift Serverless automatically provisions data warehouse capacity and intelligently scales the underlying resources. Amazon Redshift Serverless adjusts capacity in seconds to deliver consistently high performance and simplified operations for even the most demanding and volatile workloads.

With Amazon Redshift Serverless, you can benefit from the following features:

- Access and analyze data without the need to set up, tune, and manage Amazon Redshift provisioned clusters.
- Use the superior Amazon Redshift SQL capabilities, industry-leading performance, and data-lake integration to seamlessly query across a data warehouse, a data lake, and operational data sources.
- Deliver consistently high performance and simplified operations for the most demanding and volatile workloads with intelligent and automatic scaling.
- Use workgroups and namespaces to organize compute resources and data with granular cost controls.
- Pay only when the data warehouse is in use.

With Amazon Redshift Serverless, you use a console interface to reach a serverless data warehouse or APIs to build applications. Through the data warehouse, you can access your Amazon Redshift managed storage and your Amazon S3 data lake.

This video shows you how Amazon Redshift Serverless makes it easy to run and scale analytics without having to manage data warehouse infrastructure:

Amazon Redshift Serverless console

To get started with using the Amazon Redshift Serverless console, watch the following video: Getting Started with Amazon Redshift Serverless.

Serverless dashboard

On the **Serverless dashboard** page, you can view a summary of your resources and graphs of your usage.

- **Namespace overview** – This section shows the amount of snapshots and datashares within your namespace.
- **Workgroups** – This section shows all of the workgroups within Amazon Redshift Serverless.
- **Queries metrics** – This section shows query activity for the last one hour.
- **RPU capacity used** – This section shows capacity used for the last one hour.
- **Free trial** – This section shows the free trial credits remaining in your AWS account. This covers all usage of Amazon Redshift Serverless resources and operations, including snapshots, storage, workgroup, and so on, under the same account.
- **Alarms** – This section shows the alarms you configured in Amazon Redshift Serverless.

Data backup

On the **Data backup** tab you can work with the following:

- **Snapshots** – You can create, delete, and manage snapshots of your Amazon Redshift Serverless data. The default retention period is `indefinitely`, but you can configure the retention period to be any value between 1 and 3653 days. You can authorize AWS accounts to restore namespaces from a snapshot.
- **Recovery points** – Displays the recovery points that are automatically created so you can recover from an accidental write or delete within the last 24 hours. To recover data, you can restore a recovery point to any available namespace. You can create a snapshot from a recovery point if

you want to keep a point of recovery for a longer time period. The default retention period is `indefinitely`, but you can configure the retention period to be any value between 1 and 3653 days.

Data access

On the **Data access** tab you can work with the following:

- **Network and security** settings – You can view VPC-related values, AWS KMS encryption values, and audit logging values. You can update only audit logging.
- **AWS KMS key** – The AWS KMS key used to encrypt resources in Amazon Redshift Serverless.
- **Permissions** – You can manage the IAM roles that Amazon Redshift Serverless can assume to use resources on your behalf. For more information, see Identity and access management in Amazon Redshift Serverless.
- **Redshift-managed VPC endpoints** – You can access your Amazon Redshift Serverless instance from another VPC or subnet. For more information, see Connecting to Amazon Redshift Serverless from other VPC endpoints.

Limits

On the **Limits** tab, you can work with the following:

- **Base capacity in Redshift processing units (RPUs)** settings – You can set the base capacity used to process your workload. To improve query performance, increase your RPU value.
- **Usage limits** – The maximum compute resources that your Amazon Redshift Serverless instance can use in a time period before an action is initiated. You limit the amount of resource Amazon Redshift Serverless uses to run your workload. Usage is measured in Redshift Processing Unit (RPU) hours. An RPU hour is the number of RPUs used in an hour. You determine an action to occur when you reach a limit that you set, as follows:
  - Send an alert.
  - Log an entry to a system table.
  - Turn off user queries.

  You can set up to four limits.
- **Query limits** – You can add a limit to monitor performance and limits. For more information about query monitoring limits, see WLM query monitoring rules.

For more information, see Compute capacity for Amazon Redshift Serverless.

Datashares

On the **Datashares** tab you can work with the following:

- **Datashares created in my namespace** settings – You can create a datashare and share it with other namespaces and AWS accounts.
- **Datashares from other namespaces and AWS accounts** – You can create a database from a datashare from other namespace and AWS accounts.

For more information about data sharing, see Data sharing in Amazon Redshift Serverless.

Query and database monitoring

On the **Query and database monitoring** page, you can view graphs of your **Query history** and **Database performance**.

On the **Query history** tab, you see the following graphs (you can choose between **Query list** and **Resource metrics**):

- **Query runtime** – This graph shows which queries are running in the same timeframe. Choose a bar in the graph to view more query execution details.
- **Queries and loads** – This section lists queries and loads by **Query ID**.
- **RPU capacity used** – This graph shows overall capacity in Redshift Processing Units (RPUs).
- **Database connections** – This graph shows the number of active database connections.

Database performance

On the **Database performance** tab, you see the following graphs:

- **Queries completed per second** – This graph shows the average number of queries completed per second.
- **Queries duration** – This graph shows the average amount of time to complete a query.
- **Database connections** – This graph shows the number of active database connections.
- **Running queries** – This graph shows the total number of running queries at a given time.
- **Queued queries** – This graph shows the total number of queries queued at a given time.

- **Query run time breakdown** – This graph shows the total time queries spent running by query type.

Resource monitoring

On the **Resource monitoring** page, you can view graphs of your consumed resources. You can filter the data based on several facets.

- **Metric filter** – You can use metric filters to select filters for a specific workgroup, as well as choose the time range and time interval.
- **RPU capacity used** – This graph shows the overall capacity in Redshift processing units (RPUs).
- **Compute usage** – This graph shows the usage of RPU hours by period for the selected time range. For time ranges of less than 6 hours, RPU hours are shown in exact time. For time ranges of 6 hours or more, RPU hours are shown as averages.
- **Extra compute for automatic optimizations charged seconds** – This graph shows the number of RPU-seconds charged for automatic database optimizations for the selected time range. You're charged for automatic optimizations when you Amazon Redshift uses extra compute resources to run them. For more information, see Allocating extra compute resources for automatic database optimization .

Datashares

On the **Datashares** page, you can manage datashares **In my account** and **From other accounts**. For more information about data sharing, see Data sharing in Amazon Redshift Serverless.

AWS Glue Data Catalog

In the **AWS Glue Data Catalog** tab, you can view the registration status of your namespace to the AWS Glue Data Catalog. This tab only appears after you've started the registering process. For more information about registering namespaces to the AWS Glue Data Catalog, see Apache Iceberg compatibility for Amazon Redshift in the Amazon Redshift Database Developer Guide.

Considerations when using Amazon Redshift Serverless

For a list of AWS Regions where the Amazon Redshift Serverless is available, see the endpoints listed for Redshift Serverless API in the *Amazon Web Services General Reference*.

Some resources used by Amazon Redshift Serverless are subject to quotas. For more information, see Quotas for Amazon Redshift Serverless objects.

When you DECLARE a cursor, the result-set size specifications for Amazon Redshift Serverless is specified in DECLARE. Amazon Redshift Serverless has a cursor maximum total result set size of 150,000 MB.

Online Patching – Amazon Redshift Serverless offers automatic software updates without requiring traditional maintenance windows. When a new update is available, the system applies it within 14 days of release during idle periods. The update process typically takes up to 15 minutes. If no 15-minute idle period occurs within 14 days, your Serverless endpoint may experience brief unavailability. During this time, application connections to endpoints may fail. You can monitor Redshift patch releases in the "Cluster versions for Amazon Redshift" documentation. For information about Amazon Redshift Serverless SLAs, see Amazon Redshift Service Level Agreement.

Track – When Amazon Redshift releases a new workgroup version, your workgroup is updated automatically. You can control whether your workgroup is updated to the most recent release or to the previous release. For information about tracks, see Tracks for Amazon Redshift provisioned clusters and serverless workgroups.

Availability Zone IDs – When you configure your Amazon Redshift Serverless instance, open **Additional considerations**, and make sure that the subnet IDs provided in **Subnet** contain at least two of the supported Availability Zone IDs.

- For workgroups without Enhanced VPC Routing (EVR), you need two Availability Zones (AZs).
- For workgroups with EVR, you need three AZs.

To see the subnet to Availability Zone ID mapping, go to the VPC console and choose **Subnets** to see the list of subnet IDs with their Availability Zone IDs. Verify that your subnet is mapped to a supported Availability Zone ID. To create a subnet, see Create a subnet in your VPC in the *Amazon VPC User Guide*.

Two subnets (without EVR) – You must have at least two subnets, and they must span across two Availability Zones.

Three subnets (with EVR ONLY) – You must have at least three subnets when you use EVR, and they must span across three or more Availability Zones.

Free IP address requirements – When using Redshift Serverless without enhanced VPC routing (EVR) enabled, you must have at least three free IP addresses available in each subnet. This is a requirement of the proper functioning of the service.

When updating the RPUs for Redshift Serverless deployment, at least three free IP addresses must be available in each subnet to accommodate the service's operational requirements.

For more information about allocating IP addresses and understanding IP addressing in Amazon VPC, see IP addressing for your VPCs and subnets in the *Amazon VPC User Guide*.

Without EVR

If you don't use enhanced VPC routing, you must have at least three free IP addresses for each subnet, regardless of the size of the base RPU (4 to 1024 RPUs), or the RPU usage of your workgroup or workgroups. The need for 3 IP address is also applicable to workgroups that have AI-driven scaling and optimization capabilities enabled.

With Enhanced VPC Routing (EVR)

If you use enhanced VPC routing with Redshift Serverless, the minimum number of IP addresses required when creating a workgroup are as follows:

Redshift Processing Units (RPUs)	Free IP addresses required	Minimum CIDR size
4	9	/27
8	9	/27
16	13	/27
32	13	/27
64	21	/27
128	37	/26
256	69	/25
512	133	/24
1024	261	/23

With EVR, you also need free IP addresses when updating your workgroup to use more RPUs. The number of free IP addresses required when updating the subnets for a workgroup are as follows:

Redshift Processing Units (RPUs)	Updated Redshift Processing Units (RPUs)	Free IP addresses required
4	8	10
8	16	10
16	32	13
32	64	16
64	128	28
128	256	52
256	512	100
512	1024	197

Note

The maximum base RPU capacity of 1024 is only available in the following AWS Regions:

- US East (N. Virginia)
- US East (Ohio)
- US West (Oregon)
- Europe (Ireland)
- Europe (London)

For more information on allocating IP addresses, see IP addressing in the *Amazon VPC User Guide*.

Storage space after migration – When migrating small Amazon Redshift provisioned clusters to Amazon Redshift Serverless, you might see an increase in storage-space allocation after migration. This is a result of optimized storage-space allocation, resulting in preallocated storage space. This space is used over a period of time as data grows in Amazon Redshift Serverless.

Datasharing between Amazon Redshift Serverless and Amazon Redshift provisioned clusters – When datasharing where Amazon Redshift Serverless is the producer and a provisioned cluster is the consumer, the provisioned cluster must have a cluster version later than 1.0.38214. If you use a cluster version earlier than this, an error occurs when you run a query. You can view the cluster version on the Amazon Redshift console on the **Maintenance** tab. You can also run `SELECT version();`.

Max query execution time – Elapsed execution time for a query, in seconds. Execution time doesn't include time spent waiting in a queue. If a query exceeds the set execution time, Amazon Redshift Serverless stops the query. Valid values are 0–86,399.

Migrating for tables with interleaved sort keys – When migrating Amazon Redshift provisioned clusters to Amazon Redshift Serverless, Redshift converts tables with interleaved sort keys and DISTSTYLE KEY to compound sort keys. The DISTSTYLE doesn't change. For more information on distribution styles, see Working with data distribution styles in the Amazon Redshift Developer Guide. For more information on sort keys, see Working with sort keys.

VPC sharing – You can create Amazon Redshift Serverless workgroups in a shared VPC. If you do so, we recommend that you don't delete the resource share as it can result in the workgroup becoming unavailable.

IPv6 Support:

Amazon Redshift Serverless supports configuring your Amazon Redshift workgroups with both IPv4 and IPv6 addresses (dual-stack) or IPv4-only configurations within your AWS Virtual Private Clouds (VPCs). You can enable IPv6 support when creating new Amazon Redshift Serverless workgroups or modify existing workgroups to support IPv6 addressing. With this capability, you can deploy Amazon Redshift Serverless warehouses in IPv6-enabled VPC subnets and configure network settings to support the expanding address space requirements of your applications. Your applications can now communicate with Amazon Redshift Serverless warehouses using either IPv4 or IPv6 protocols, ensuring compatibility with both existing and future network architectures.

Compute capacity for Amazon Redshift Serverless

With Amazon Redshift Serverless compute capacity scales automatically up and down to match your workload requirements. Compute capacity refers to the processing power and memory allocated to your Amazon Redshift Serverless workloads. Common use cases include handling peak traffic periods, running complex analytics, or processing large volumes of data efficiently. The following terms provide details on how Amazon Redshift manages compute capacity.

RPUs

Amazon Redshift Serverless measures data warehouse capacity in Redshift Processing Units (RPUs). RPUs are resources used to handle workloads. One RPU provides 16 GB of memory.

Base capacity

This setting specifies the base data warehouse capacity Amazon Redshift uses to serve queries. Base capacity is specified in RPUs. You can set a base capacity in Redshift Processing Units (RPUs). Setting a higher base capacity ensures improves query performance , especially for data processing jobs that require a lot of resources. The default base capacity for Amazon Redshift Serverless is 128 RPUs. You can adjust the **Base capacity** setting from 4 RPUs to 512 RPUs. You can set this value to 4 RPUs, or in units of 8 at or above 8 RPUs (8,16,24...512). You can set this value using the AWS console, the `UpdateWorkgroup` API operation, or `update-workgroup` operation in the AWS CLI.

With a minimum base capacity of 4 RPU, you have flexibility to run simpler to more complex workloads based on your data warehouse cost and capacity requirements. The 4 base RPU capacity is targeted towards warehouses that contain less than 32TB of data, and the 8, 16, and 24 RPU base RPU capacities are targeted towards workloads that require less than 128TB of data. If your data requirements are greater than 128 TB, you must use a minimum of 32 base RPUs. In addition, for workloads that have tables with large number columns and higher concurrency, we recommend using 32 or more base RPUs.

The maximum base RPUs available, 1024, adds the highest level of computing resources to your workloads. This provides more flexibility to support workloads of large complexity and accelerates loading and querying data.

> **Note**
>
> An expanded maximum base RPU capacity of 1024 is available in the following AWS Regions. In other regions, the maximum base capacity is 512 RPUs.

- US East (N. Virginia)
- US East (Ohio)
- US West (Oregon)
- Europe (Ireland)
- Europe (Frankfurt)

You can increment or decrement RPUs in units of 32 when setting a base capacity between 512-1024.

If you manage larger and more complex workloads, consider increasing the size of your Redshift Serverless data warehouse. Larger warehouses have access to more compute resources, allowing them to process queries more efficiently.

Following are some instances where having a higher base capacity is beneficial:

- You have complex queries that take a long time to run
- Your tables have a large number of columns.
- Your queries have a high number of JOINs.
- Your queries aggregate or scan large amounts of data from an external source, such as a data lake.

For more information about Amazon Redshift Serverless quotas and limits, go to Quotas for Amazon Redshift Serverless objects.

Considerations and limitations for Amazon Redshift Serverless capacity

The following are considerations and limitations for Amazon Redshift Serverless capacity. For general Redshift Serverless considerations, see Considerations when using Amazon Redshift Serverless.

- Configurations of 4 base RPUs support managed storage capacity of up to 32 TB. If you're using more than 32 TB of managed storage, you can't set the base RPU to less than 8 RPUs.
- Configurations of 8 or 16 base RPUs support Redshift managed storage capacity of up to 128 TB. If you're using more than 128 TB of managed storage, you can't set the base to less than 32 RPU.

- Editing your workgroup's base capacity might cancel some of the queries running on your workgroup.
- Redshift Serverless scales RPUs for your data warehouse using these increments:
  - 4 to 8 RPUs: Increases in steps of 4 RPUs.
  - 8 to 512 RPUs: Increases in steps of 8 RPUs.
  - 512 to 1024 RPUs: Increases in steps of 32 RPUs.
- Vacuum boost is supported only for 8 RPUs and Above. For 8 RPUs and less, use the following command instead:

```
VACUUM [FULL | SORT ONLY | DELETE ONLY | REINDEX | RECLUSTER] [table_name] [TO
 threshold PERCENT]
```

Redshift Serverless with 4 Redshift Processing Units (RPUs) capacity

Redshift Serverless with 4 base RPUs capacity is ideal for smaller or less demanding workloads. This entry point offers a flexible and cost-effective solution. This entry-level configuration supports data warehouses with up to the following resources:

- Up to 32 TB of Redshift managed storage.
- A maximum of 100 columns per table
- 64 GB of memory

If you need to exceed these limitations, you must increase your base capacity manually, rather than relying on auto-scaling. Once you scale your data warehouse beyond 4 RPUs, your data warehouse will continue to use more RPUs, and Amazon Redshift won't scale your data warehouse back down to 4 RPUs.

> **Note**
>
> You can create tables with more than 100 columns when using 4 base RPUs, however, we recommend that you limit tables to 100 columns. Exceeding this limit may cause your data warehouse to exhaust its memory during query execution, which decreases performance.

You can create data warehouses that use 4 RPUs in the following AWS Regions:

- US East (Ohio)
- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- Asia Pacific (Mumbai)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Europe (Ireland)
- Europe (Stockholm)

AI-driven scaling and optimization

> **Note**
>
> AI-driven scaling and optimization is available in all AWS Regions where Amazon Redshift Serverless is available.

Amazon Redshift Serverless offers an advanced AI-driven scaling and optimization feature to meet diverse workload requirements. Data warehouses may have the following provisioning issues:

- Data warehouses may be over-provisioned to improve performance of resource-intensive queries
- Data warehouses may be under-provisioned to save costs.

Striking the right balance between performance and cost for data warehouse workloads is challenging, especially with ad-hoc queries and growing data volumes. When running mixed workloads, comprising both low and high resource-intensive queries, there is a need for intelligent scaling. The AI-driven scaling and optimization feature automatically scales Serverless compute or RPUs in response to data growth. This feature also helps maintain query performance within targeted price-performance objectives. The AI-driven scaling and optimization dynamically allocates compute resources as data volumes increase, ensuring queries continue to meet performance targets. AI-driven scaling and optimization allows the service to adapt seamlessly to

changing workload requirements, without the need for manual intervention or complex capacity planning.

Amazon Redshift Serverless provides a more comprehensive and responsive scaling solution based on factors such as query complexity and data volume. This feature allows for optimizing workload price-performance while maintaining the flexibility to handle varying workloads and growing datasets efficiently. Amazon Redshift Serverless can automatically apply AI-driven scaling and optimization to your Amazon Redshift Serverless endpoint to meet your specified price-performance targets for your Serverless workgroup. This automatic price-performance optimization is especially helpful if you don't know what base capacity to set for your workloads, or if some parts of your workload might benefit from more allocated resources.

Example

If your organization typically runs workloads that only require 32 RPU but suddenly introduces a more complex query, you might not know the appropriate base capacity. Setting a higher base capacity yields better performance but also incurs higher costs, so the cost might not match your expectations. Using AI-driven scaling and optimization, Amazon Redshift Serverless automatically adjusts the RPUs to meet your price-performance targets while keeping costs optimized for your organization. This automatic optimization is useful regardless of workload size. The automatic optimization can help you meet your organization's price-performance targets if you have any number of complex queries.

> **Note**
>
> Price-performance targets are a workgroup-specific setting. Different workgroups can have different price-performance targets.

To configure price-performance targets, use the AWS console. The price-performance target is enabled by default for all new Serverless workgroups and is set to **Balanced**. You can modify the price-performance target after you create the Serverless workgroup. .

To edit the price-performance target for your workgroup

1. In the Amazon Redshift Serverless console, choose **Workgroup configuration**.
2. Choose the workgroup for which you want to edit the price-performance target. Choose the **Performance** tab, then choose **Edit**.

3. Choose **Price-performance** target, and adjust the slider to your desired setting.
4. Choose **Save changes**.
5. To update the maximum amount of RPUs that Amazon Redshift Serverless can allocate to your workload, choose the **Limits** tab of the **Workgroup Configuration** section.

You can use the **Price-performance target** slider to set your desired balance between cost and performance. By moving the slider, you can choose one of the following options:

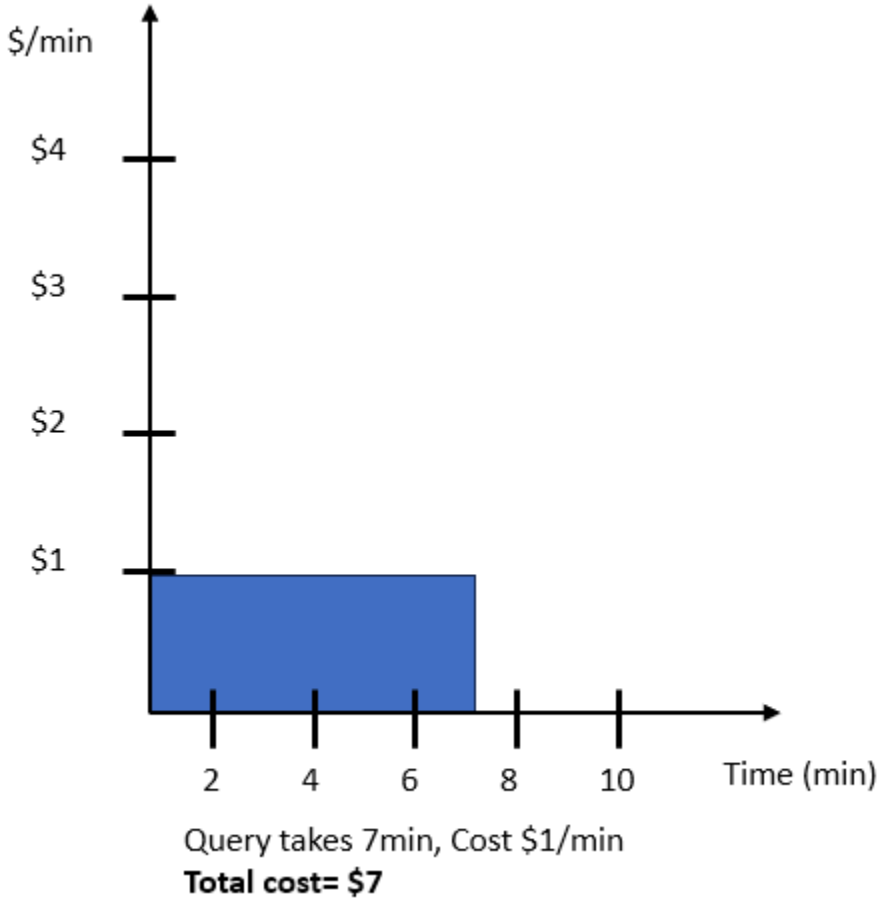
- **Optimizes for cost** — This setting prioritizes cost savings. Amazon Redshift Serverless attempts to scale up compute capacity automatically only when doing so is likely to reduce cost. Amazon Redshift Serverless also attempts to scale down compute resources for lower cost, possibly increasing query runtimes.
- **Balanced** — This setting creates a balance between performance and cost. Amazon Redshift Serverless scales for performance, and might result in a moderate cost increase or decrease. This is the recommended setting for most Amazon Redshift Serverless data warehouses.
- **Optimizes for performance** — This setting prioritizes performance. Amazon Redshift scales aggressively for high performance, potentially incurring higher costs.
- Intermediate positions: You can also set the slider to one of two intermediate positions between **Balanced** and **Optimizes for cost** or **Optimizes for performance**. Use these settings if full optimization for cost or performance is too extreme.

Considerations when choosing your price-performance target

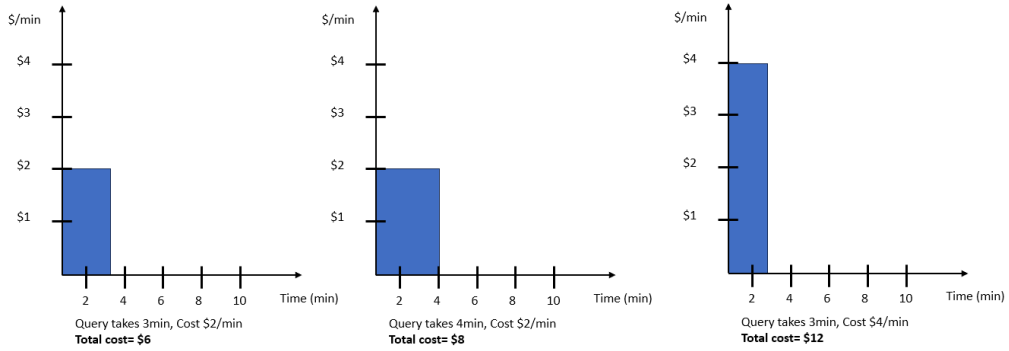
You can use the price-performance slider to choose your desired price-performance target for your workload. The AI-driven scaling and optimization algorithm learns over time from your workload history, and improves prediction and decision accuracy.

Example

For this example, assume a query that takes seven minutes and costs $7. The following figure shows the query runtimes and cost with no scaling.



A given query might scale in a few different ways, as shown below. Based on the price-performance target you choose, AI-driven scaling and optimization predicts how the query trades off performance and cost, and scales it accordingly. Choosing the different slider options yields the following results:



- **Optimizes for Cost** — With the **Optimizes for Cost** option, your data warehouse scales favoring choices that lower your costs. In the preceding example, the super linear scaling approach demonstrates this behavior. Scaling will only occur if it can be done in a cost-effective manner according to the scaling model predictions. If the scaling models predict that cost-optimized scaling isn't possible for the given workload, then the data warehouse won't scale.
- **Balanced** — With the **Balanced** option, the system scales while balancing both cost and performance considerations, with a potential limited increase in cost. The **Balanced** option performs superlinear, linear, and possibly sublinear workload scaling.
- **Optimizes for Performance** — With the **Optimizes for Performance** option, in addition to the previous methods for improving performances, the system also scales even if the costs are higher, and possibly not proportional to the runtime improvement. With **Optimizes for Performance**, the system performs superlinear scaling, linear scaling, and sublinear scaling if possible. The closer the slider position is to the **Optimizes for Performance** position, the more Amazon Redshift Serverless permits sublinear scaling.

Note the following when setting the **Price-Performance** slider:

- You can change the price-performance setting at any time, but the workload scaling won't change immediately. The scaling changes over time as the system learns about the current workload. We suggest monitoring a Serverless Workgroup for 1-3 days to verify the impact of the new setting.
- The price-performance slider options **Max capacity** and **Max RPU-hours** work together. **Max capacity** and **Max RPU-hours** are the controls to limit the maximum RPUs that Amazon Redshift Serverless allows the data warehouse to scale, and the maximum RPU hours that Amazon Redshift Serverless allows the data warehouse to consume. Amazon Redshift Serverless always honors and enforces these settings, regardless of the price-performance target setting.

Monitoring resource autoscaling

You can monitor AI-driven scaling and optimization in the following ways:

- Review the RPU capacity used graph on the Amazon Redshift console.
- Monitor the `ComputeCapacity` metric under `AWS/Redshift-Serverless` and `Workgroup` in CloudWatch.
- Query the SYS_QUERY_HISTORY view. Provide the specific query ID or query text to identify the time period. Use this time period to query the SYS_SERVERLESS_USAGE system view to find the

compute_capacity value. The compute_capacity field shows the RPUs scaled during the query runtime.

Use the following example to query the SYS_QUERY_HISTORY view. Replace the sample value with your query text.

```
select query_id,query_text,start_time,end_time, elapsed_time/1000000.0
 duration_in_seconds
from sys_query_history
where query_text like '<query_text>'
and query_text not like '%sys_query_history%'
order by start_time desc
```

Run the following query to see how compute_capacity scaled during the period from start_time to end_time. Replace start_time and end_time in the following query with the output of the preceding query:

```
select * from sys_serverless_usage
where end_time >= 'start_time'
and end_time <= DATEADD(minute,1,'end_time')
order by end_time asc
```

For step-by-step instructions for using these features, see Configure monitoring, limits, and alarms in Amazon Redshift Serverless to keep costs predictable .

Considerations when using AI-driven scaling and optimization

Consider the following when using AI-driven scaling and optimization:

- For existing workloads on Amazon Redshift Serverless requiring 8 to 512 Base RPU, use AI-driven scaling and optimization for optimal results. We do not recommend using this feature for workloads below or above this RPU range.
- Price-performance targets automatically optimize the workload, though results might vary. Use this feature over time so the system can learn your specific patterns by running a representative workload.
- AI-driven scaling and optimization uses optimal times to apply optimizations to Serverless workgroups based on the workload running on your Amazon Redshift Serverless instance.

To learn more about AI-driven scaling and optimization, watch the following video.

Billing for Amazon Redshift Serverless

Billing for compute capacity

You can purchase capacity for Amazon Redshift Serverless in two ways:

- **You can purchase on-demand capacity** – When you choose on-demand compute capacity, you pay for resources as you go. This is the best choice if you're just beginning to use Amazon Redshift Serverless or if you don't have a good sense yet of your steady usage patterns. On-demand offers the most flexibility. For more information, see Billing for on-demand compute capacity.
- **You can purchase reservations** – A reservation provides a discount when you buy a preset amount of compute resources for a specific amount of time, for example for a year. It's a good idea when you know you're going to use an amount of capacity steadily. It's helpful for saving money when you can forecast some of your capacity needs. For more information, see Billing for serverless reservations.

You can use reservations and on-demand resources together. It isn't required that you use one or the other.

For detailed pricing information, see Amazon Redshift pricing.

Billing for on-demand compute capacity

Base capacity and its effect on billing

When queries run, you're billed according to the capacity used in a given duration, in RPU hours on a per-second basis. When no queries are running, you aren't billed for compute capacity. You are also charged for Redshift Managed Storage (RMS), based on the amount of data stored.

When you create your workgroup, you have the option to set the **Base capacity** for computing. To meet the price/performance requirements of your workload at a workgroup level, adjust the base capacity higher or lower for an existing workgroup. Select the workgroup from **Workgroup configuration** and choose the **Limits** tab to change the base capacity using the console.

As the number of queries increases, Amazon Redshift Serverless scales automatically to provide consistent performance.

Maximum RPU hours usage limit

To keep costs predictable for Amazon Redshift Serverless, you can set the **Maximum RPU hours** used per day, per week, or per month. You can set it using the console or with the API. When a limit is reached, you can specify that a log entry is written to a system table, or you receive an alert, or user queries are turned off. Setting the maximum RPU hours helps keep your cost under control. Settings for maximum RPU hours apply to your workgroup for both queries that access data in your data warehouse and queries that access external data, such as in an external table in Amazon S3.

The following is an example:

Assume you set a limit for 100 hours for each week. To do this on the console, you do the following:

1. Choose your workgroup and then choose **Manage usage limits** under the **Limits** tab.
2. Add a usage limit, choosing the **Weekly** frequency, a duration of **100** hours, and the setting the action to **Turn off user queries**.

In this example, if you reach the 100 RPU hour limit for a week, queries are turned off.

Setting the maximum RPU hours for the workgroup doesn't limit the performance or compute resources for the workgroup. You can adjust the settings at any time with no affect on query processing. The goal for setting maximum RPU hours is to help you meet your price and performance requirements. For more information about serverless billing, see Amazon Redshift pricing.

Another way to keep the cost for Amazon Redshift Serverless predictable is to use AWS Cost Anomaly Detection to reduce the chance for billing surprises and provide more control.

> **Note**
>
> The Amazon Redshift pricing calculator is helpful for estimating pricing. You enter the compute resources you need and it provides a preview of the cost.

Setting Max capacity to control costs for compute resources

The Max capacity setting serves as the RPU ceiling that Amazon Redshift Serverless can scale up to. It helps control your cost for compute resources. In a similar way to how base capacity sets

a minimum amount of compute resources available, Max capacity sets a ceiling on RPU usage. That way, it helps your spending adhere to your plans. Max capacity applies specifically to each workgroup and it limits compute usage at all times.

How Max capacity differs from RPU hour usage limits

Both maximum RPU hour limits and Max capacity can help you control cost, but they work in different ways. The following points explain the difference:

- *Max capacity* – This setting establishes the highest count of RPUs that Amazon Redshift Serverless can use for automatic scaling purposes.
- *Maximum RPU hours usage limit* – Unlike Max capacity, this setting doesn't set a ceiling on capacity. But it does perform other actions to help you limit costs. These include adding an entry to a log, notifying you, or stopping queries from running, if you choose.

You can use Max capacity exclusively, or you can complement it with actions from maximum RPU hour usage limits.

A Max capacity use case

Each workgroup can have a different Max capacity setting. It helps you enforce budgeting requirements. To illustrate how this works, assume the following:

- You have a workgroup with the base capacity set to 256 RPUs. You have steady workloads at just over 256 RPUs for most of the month.
- Max capacity is set to 512 RPUs.

Assume you have unexpected high use over a three-day period to generate ad-hoc statistical reports. In this case, you have Max capacity set to avoid compute costs beyond those for 512 RPUs. When you do this, you can be sure that compute capacity won't exceed this upper bound.

Usage notes for Max capacity

These notes can help you set Max capacity appropriately:

- Each Amazon Redshift Serverless workgroup can have a different Max capacity setting.
- If you have a period of very high resource usage and Max capacity is set to a low RPU level, it can delay workload processing and result in a user experience that isn't optimal.

- You can view capacity used over a period of time on the Amazon Redshift Serverless dashboard. For more information about viewing summary data, see Checking Amazon Redshift Serverless summary data using the dashboard.
- Max capacity changes usually do not disrupt running queries. However, if you set Max capacity below your current base capacity, Amazon Redshift Serverless will immediately reduce capacity. This can affect your running workload.
- The top Max capacity setting is 5632 RPUs.

How to set Max capacity

You can set Max capacity in the console. For an existing workgroup, you can change the setting under **Workgroup configuration**. You can also use the CLI to set it by using a command like the following sample:

```
aws redshift-serverless update-workgroup --workgroup-name myworkgroup --max-capacity
 512
```

This sets the Max capacity setting for the workgroup with the given name. After setting it, you can check the value on the console to verify it. You can also check the value using the CLI by running the `get-workgroup` command.

You can turn off the Max capacity setting by setting it to `-1`, like the following:

```
aws redshift-serverless update-workgroup --workgroup-name myworkgroup --max-capacity -1
```

Monitoring Amazon Redshift Serverless usage and cost

There are several ways you can estimate usage and billing for Amazon Redshift Serverless. System views can be helpful because the system metadata, including query and usage data, is timely and you don't have to do any setup to query it. CloudWatch can also be useful for monitoring usage for your Amazon Redshift Serverless instance, and has additional features to provide insights and set actions.

Visualizing usage by querying a system view

Query the SYS_SERVERLESS_USAGE system table to track usage and get the charges for queries:

```
select trunc(start_time) "Day",
(sum(charged_seconds)/3600::double
```

```
precision) * <Price for 1 RPU> as cost_incurred
from sys_serverless_usage
group by 1
order by 1
```

This query provides the cost per day incurred for Amazon Redshift Serverless, based on usage.

Usage notes for determining usage and cost

- You pay for the workloads you run in RPU-hours on a per-second basis, with a 60-second minimum charge.
- Records from the sys_serverless_usage system table show cost incurred in 1-minute time intervals. Understanding the following columns is important:

  The charged_seconds column:

  - Provides the compute unit (RPU) seconds that were charged during the time interval. The results include any minimum charges in Amazon Redshift Serverless.
  - Has information about compute-resource usage after transactions complete. Thus, this column value may be 0 if transactions haven't finished.

  The compute_seconds column:

  - Provides real-time compute usage information. This doesn't include any minimum charges in Amazon Redshift Serverless. Thus it can differ to some degree from the charged seconds billed during the interval.
  - Shows usage information during each transaction (even if a transaction hasn’t ended), and hence the data provided is real-time.
- There are situations where compute_seconds is 0 but charged_seconds is greater than 0, or vice versa. This is normal behavior resulting from the way data is recorded in the system view. For a more accurate representation of serverless usage details, we recommend aggregating the data in SYS_SERVERLESS_USAGE.

For more information about monitoring tables and views, see Monitoring queries and workloads with Amazon Redshift Serverless.

Visualizing usage with CloudWatch

You can use the metrics available in CloudWatch to track usage. The metrics generated for CloudWatch are `ComputeSeconds`, indicating the total RPU seconds used in the current minute

and `ComputeCapacity`, indicating the total compute capacity for that minute. Usage metrics can also be found on the Redshift console on the Redshift **Serverless dashboard**. For more information about CloudWatch, see What is Amazon CloudWatch?

Billing for serverless reservations

Amazon Redshift Serverless allows you to run and scale analytics without having to provision and manage clusters with a pay-as-you-go pricing model. Now with serverless reservations, you can further optimize your compute costs and improve cost predictability of existing and new workloads on Redshift Serverless.

Amazon Redshift manages serverless reservations at the AWS payer account level, and reservations can be shared between multiple AWS accounts, letting you reduce your compute costs by up to 24% on all Redshift Serverless workloads in your AWS account. Amazon Redshift bills serverless reservations hourly and meters reservations per second, offering a consistent billing model, 24 hours a day, seven days a week, while maintaining flexibility offered by Redshift Serverless. Amazon Redshift charges any usage exceeding the specified RPU level is at standard on-demand rates.

> **Note**
>
> If you want to limit on-demand usage, you can use the **Max capacity** setting to set resource-usage limits for your workgroups. For more information, see Billing for Amazon Redshift Serverless.

Benefits of serverless reservations

Serverless reservations are a discounted pricing option for Amazon Redshift Serverless. Serverless reservations give you the option to commit to a specified number of Redshift Processing Units (RPUs) for a year at a discount from on-demand (OD) rates, with no upfront payment. You can receive a greater discount with an upfront payment. With serverless reservations, you can optimize your compute costs and improve cost predictability of existing and new workloads on Serverless.

Each serverless reservation is purchased at the AWS account level and can be shared between multiple Amazon Redshift Serverless workgroups in the same payer account. This gives you flexibility in how the discount is applied. Multiple workgroups with different workload patterns can share the reservation.

How a serverless reservation works

Reserving RPUs is a simple process that takes only a few minutes to complete. It includes specifying the RPU level to reserve and the payment type. Amazon Redshift Serverless uses the standard AWS billing and cost management tool that helps you determine the reservation level you need and to monitor your usage continuously. Serverless reservations are managed at the AWS payer account level and can be shared under the same payer account, and let you reduce your compute costs by up to 24% on all Redshift Serverless workloads in your AWS account. Serverless reservations are billed hourly and metered per second, offering a consistent billing model, 24 hours a day, seven days a week, while maintaining flexibility offered by Redshift Serverless. Any usage exceeding the specified RPU level is charged at standard Redshift Serverless on-demand rates.

You can purchase multiple serverless reservations within the same AWS account. When you purchase additional serverless reservations, they layer on each other. For instance, if you purchase two reservations and choose 100 RPUs for each, it gives you a total of 200 RPUs at a discounted rate.

> **Note**
>
> If you want to set a limit for on-demand usage, you can set the maximum RPUs in the Amazon Redshift Serverless console for a workgroup by choosing the **Limits** tab and then selecting **Manage usage limits**.

After you purchase a serverless reservation, it goes into effect immediately and shows up in the Redshift console in the Serverless reservations dashboard.

Analyzing your RPU (Redshift Processing Unit) use to determine the reservation level you need

Redshift Serverless Reservations let you lock in predictable, lower compute costs by committing to a specific number of Redshift Processing Units (RPUs) for one year, giving you discounts over on-demand pricing. These discounts can be up to 20 percent with the no-upfront option, or up to 24 percent when you pay all-upfront. You purchase Redshift Serverless Reservations at the AWS payer-account level, and your savings automatically apply to any Redshift Serverless workgroup in any AWS linked account, so you can centrally manage budgets while supporting multiple teams. Redshift Serverless meters the usage at per-second granularity, averaged across each hour, and then billed hourly, ensuring you pay only for the capacity you use. Redshift Serverless Reservations

combine flexible application across accounts with term-based savings, giving you predictable analytics prices without sacrificing the agility of Redshift Serverless.

Analyzing RPU Use for Reservations

You can determine your RPU usage levels in one of two ways: You can use the Redshift Serverless dashboard for a seven-day view, or use Cost Explorer for long term analysis. The following procedures demonstrate how to analyze your RPU use:

Method 1: Redshift Serverless Dashboard (7-day view)

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. Open the Serverless dashboard.
3. Choose your workgroup.
4. View RPU capacity usage for a period in length from the last hour up to one week.

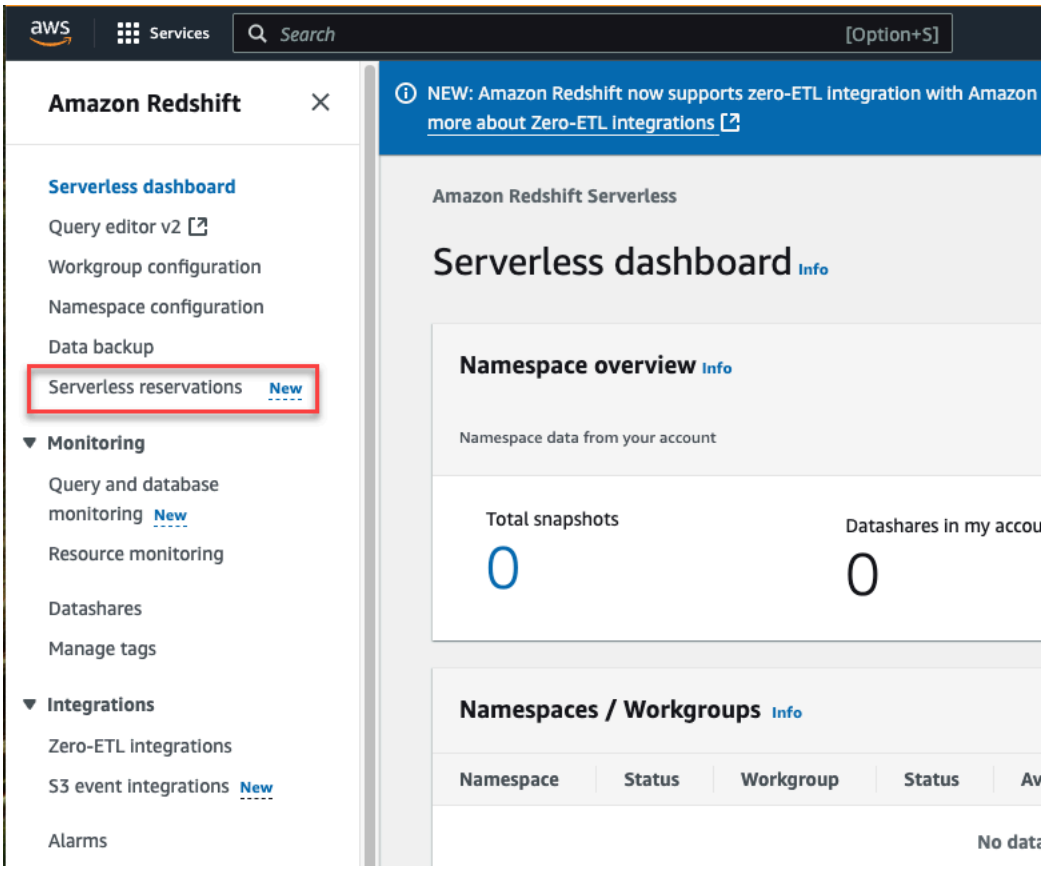
Method 2: AWS Cost Explorer (Long-term analysis)

1. Sign in to the AWS Management Console and open the Cost Explorer console at https://console.aws.amazon.com/costmanagement/.
2. Set Granularity to **Hourly**
3. Group by **Usage type**
4. Apply the following filters:
   - Service: Redshift
   - Region: Your local region
   - Usage type: Filter for **Redshift:ServerlessUsage**
5. Review the Cost and usage graph for hourly serverless usage in your chosen region

Purchasing a serverless reservation using the console

When you purchase a reservation, you choose the RPU level that will be discounted. Prior to selecting your RPU level, it's good to know your base capacity and the on-demand capacity you use over time. This section shows you how to determine your capacity and reserve a serverless reservation.

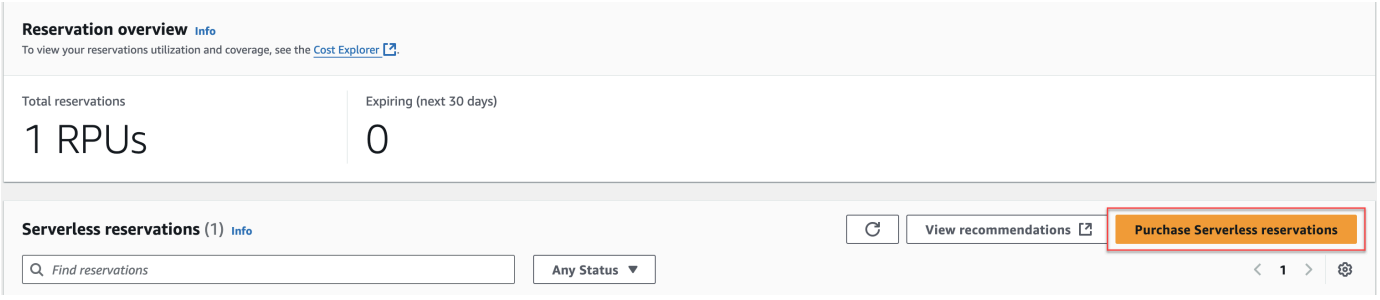
To start, in the Redshift console, choose **Serverless**, and then **Serverless reservations** from the menu.



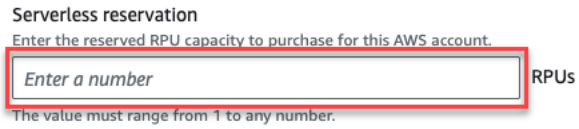
The console shows a description of the feature and a list of existing reservations. From here you can purchase a reservation, or you can use the reports and monitoring tools available to check your current usage. These help you determine your RPU levels and how many RPUs are appropriate to reserve.

To purchase a reservation, complete the following steps:

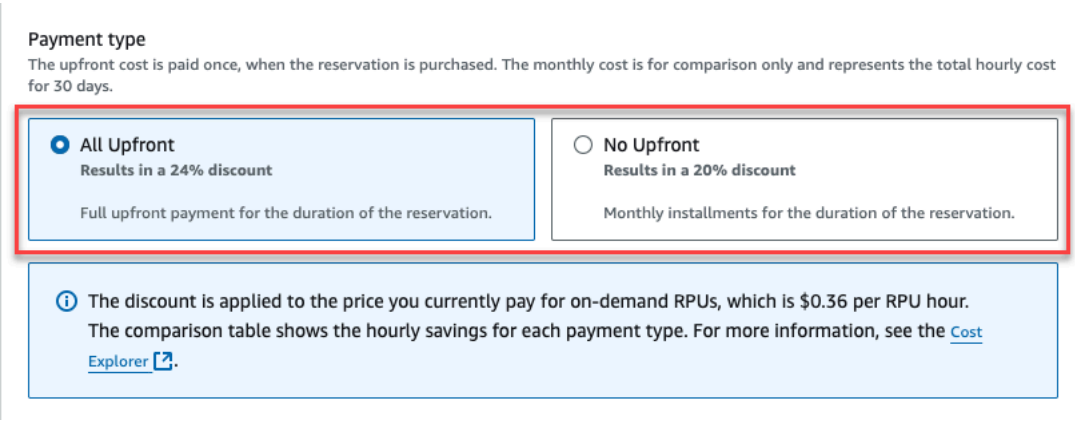
1. Choose **Purchase serverless reservations**.



2. A walk through appears, which has a series of selections. Enter the **Serverless reservation** RPU level to reserve. If you are unsure what this level should be, you can use the tools described further along in this section.

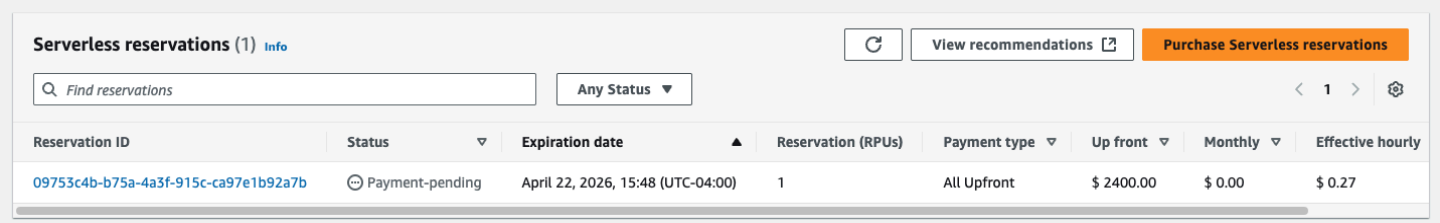


3. Set the payment type. You can choose to pay upfront for your reserved RPUs, or you can pay monthly. If you choose to pay up front, you get a bigger discount.



4. When you finish making the selections, choose **Purchase serverless reservations** and then **Confirm**.

After you confirm the reservation, it appears in the list of reservations.



Usage notes

- You can't change or delete a reservation. But you can create additional reservations to get more coverage.
- Redshift Serverless uses reserved RPUs for a workload prior to using on-demand RPUs, to ensure cost savings. If you exceed the number of RPUs that you have reserved, you begin to accrue charges for those additional RPUs at the Redshift Serverless on-demand rate.

- Free credits for Amazon Redshift Serverless aren't applied to serverless reservations, only to RPUs billed on-demand.

Serverless reservation examples

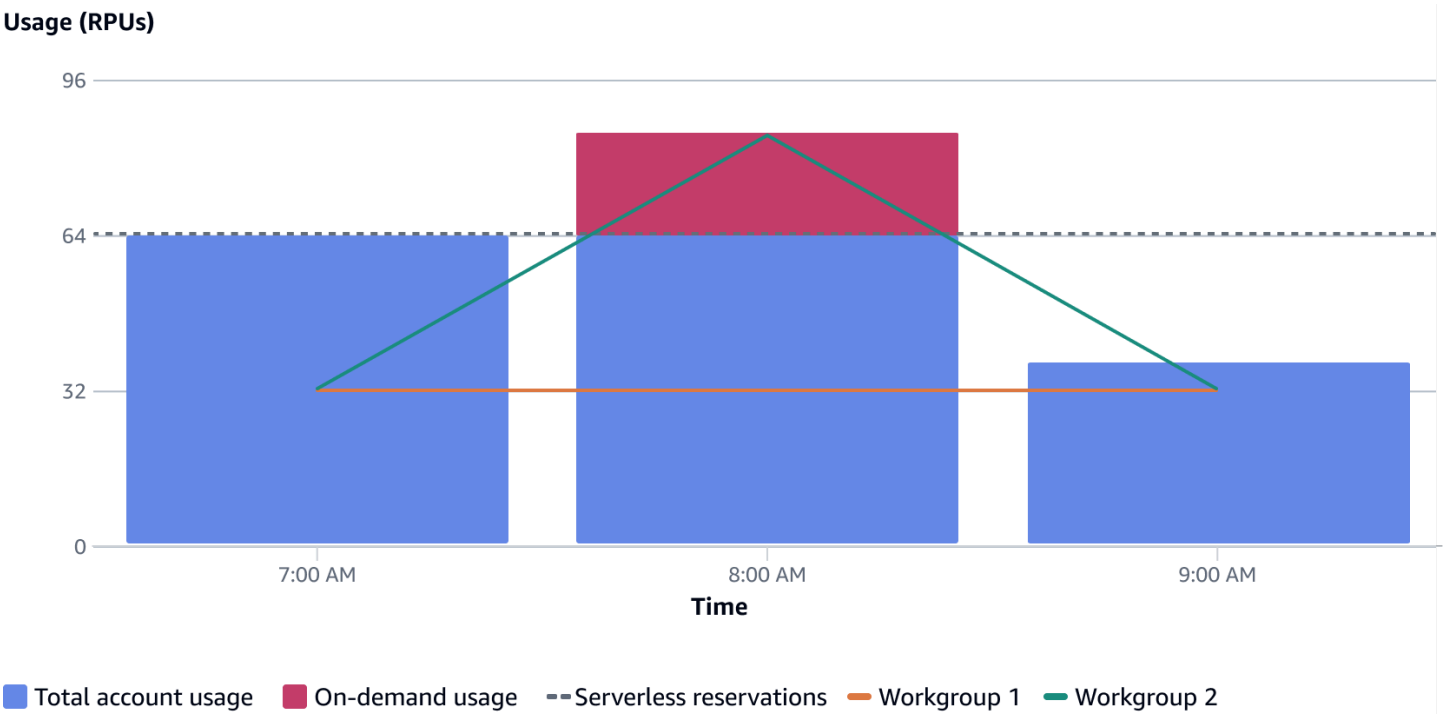
In this scenario, your AWS payer/ linked account has two Amazon Redshift workgroups:

- Workgroup 1 has steady state usage, such as for a business-intelligence team.
- Workgroup 2 has unpredictable workloads with spikes in usage, such as for ETL operations.

You want to optimize the costs for these workgroups, so you purchase a one-year serverless reservation. Based on historical data, you determine that both workgroups consume 64 RPUs at a steady state. Workgroup 2, however, occasionally increases from 32 RPUs to 48 RPUs and drops to 24 RPUs for short periods. You set the RPU level of your reservation at 64 RPUs to start, which is aligned with historical trends. The per-hour billing details are as follows:

- For the first hour, similar to historical usage trends, both workgroups use 32 RPUs for total account usage of 64 RPUs. For this hour, all the RPUs are charged at the serverless reservations discounted rate. This is because the usage level of 64 RPUs is equal to the 64 RPU serverless reservation.
- For the second hour, workgroup 1 continues to use 32 RPUs. However, workgroup 2 spikes to 48 RPUs, for a total account usage of 80 RPUs. For this hour, 64 RPUs are charged at the serverless reservations discounted rate, and 16 RPUs are charged at the Redshift Serverless on-demand rate.
- For the third hour, workgroup 1 continues to consume 32 RPUs and workgroup 2 decreases to 8 RPUs. In this hour, the account is charged at the 64 RPU serverless reservation rate, even though the account total is 40 RPU.

See the following diagram for workgroup usage evolution, and on-demand and serverless reservation rates billing details:



Purchasing a serverless reservation using the AWS CLI or Amazon Redshift API

You use `create-reservation` to create an RPU reservation. The following shows the command:

```
create-reservation
--capacity
--offering-id
```

You set `capacity` to the number of RPUs you want to reserve.

Billing for storage

Primary storage capacity is billed as Redshift Managed Storage (RMS). Storage is billed by GB / month. Storage billing is separate from billing for compute capacity. Storage used for manual snapshots is billed based on the total unique data blocks across all active manual snapshots at the standard backup billing rates. Recovery points retained for less than 24 hours are not charged.

Data transfer costs and machine learning (ML) costs apply separately, the same as provisioned clusters. Snapshot replication and data sharing across AWS Regions are billed at the transfer rates outlined on the pricing page. For more information, see Amazon Redshift pricing.

Visualizing billing usage with CloudWatch

The metric `ServerlessUniqueSnapshotStorage`, which tracks snapshot storage usage, is generated and sent to CloudWatch. For more information about CloudWatch, see What is Amazon CloudWatch?

Using the Amazon Redshift Serverless free trial

Amazon Redshift Serverless offers a free trial. If you participate in the free trial, you can view the free trial credit balance in the Redshift console, and check free trial usage in the SYS_SERVERLESS_USAGE system view. Note that billing details for free trial usage does not appear in the billing console. You can only view usage in the billing console after the free trial ends. For more information about the Amazon Redshift Serverless free trial, see Amazon Redshift Serverless free trial.

Billing usage notes

- **Recording usage** - A query or transaction is only metered and recorded after the transaction completes, is rolled back, or stopped. For instance, if a transaction runs for two days, RPU usage is recorded after it completes. You can monitor ongoing use in real time by querying `sys_serverless_usage`. Transaction recording may reflect as RPU usage variation and effect costs for specific hours and for daily use.
- **Writing explicit transactions** - It's important as a best practice to end transactions. If you don't end or roll back an open transaction, Amazon Redshift Serverless continues to use RPUs. For example, if you write an explicit `BEGIN TRAN`, it's important to have corresponding `COMMIT` and `ROLLBACK` statements.
- **Cancelled queries** - If you run a query and cancel it before it finishes, you are still billed for the time the query ran.
- **Scaling** - The Amazon Redshift Serverless instance may initiate scaling for handling periods of higher load, in order to maintain consistent performance. Your Amazon Redshift Serverless billing includes both base compute and scaled capacity at the same RPU rate.
- **Scaling down** - Amazon Redshift Serverless scales up from its base RPU capacity to handle periods of higher load. In some cases, RPU capacity can remain at a higher setting for a period after query load falls. We recommend that you set maximum RPU hours in the console to guard against unexpected cost.
- **System tables** - When you query a system table, the query time is billed.

- **Redshift Spectrum** - When you have Amazon Redshift Serverless, and you run queries, there isn't a separate charge for data-lake queries. For queries on data stored in Amazon S3, the charge is the same, by transaction time, as queries on local data.
- **Federated queries** - Federated queries are charged in terms of RPUs used over a specific time interval, in the same manner as queries on the data warehouse or data lake.
- **Storage** - Storage is billed separately, by GB / month.
- **Minimum charge** - The minimum charge is for 60 seconds of resource usage, metered on a per-second basis.
- **Snapshot billing** - Snapshot billing doesn't change. It's charged according to storage, billed at a rate of GB / month. You can restore your data warehouse to specific points in the last 24 hours at a 30 minute granularity, free of charge. For more information, see Amazon Redshift pricing.
- **Automatic optimizations run using extra compute resources** - Amazon Redshift Serverless usually runs automatic optimization operations alongside user queries. These operations are known as autonomics, and you aren't charged for them.

  If you enable allocating extra compute resources, Amazon Redshift will run autonomics when necessary even in periods of high user activity. In such cases, you can be billed for the time spent running autonomics. For more information, see Allocating extra compute resources for automatic database optimization in the *Amazon Redshift Database Developer Guide*.

Amazon Redshift Serverless best practices for keeping billing predictable

The following are best practices and built-in settings that help keep your billing consistent.

- Make sure to end each transaction. When you use `BEGIN` to start a transaction, it's important to `END` it as well.
- Use best-practice error handling to respond gracefully to errors and end each transaction. Minimizing open transactions helps to avoid unnecessary RPU use.
- Use `SESSION TIMEOUT` to help end open transactions and idle sessions. It causes any session kept idle or inactive for more than 3600 seconds (1 hour) to time out. It causes any transaction kept open and inactive for more than 21600 seconds (6 hours) to time out. This timeout setting can be changed explicitly for a specific user, such as when you want to keep a session open for a long-running query. The topic CREATE USER shows how to adjust `SESSION TIMEOUT` for a user.
  - In most cases, we recommend that you don't extend the `SESSION TIMEOUT` value, unless you have a use case that requires it specifically. If the session remains idle, with an open

transaction, it can result in a case where RPUs are used until the session is closed. This will result in unnecessary cost.

- Amazon Redshift Serverless has a maximum time of 86,399 seconds (24 hours) for a running query. The maximum period of inactivity for an open transaction is six hours before Amazon Redshift Serverless ends the session associated with the transaction. For more information, see Quotas for Amazon Redshift Serverless objects.

Amazon Redshift Serverless billing with connection pooling

Amazon Redshift Serverless treats all incoming queries as billable user activity, including lightweight health-check queries sent by connection pools. This behavior applies regardless of whether the query originates from an application, a JDBC/ODBC driver, or a connection pooling framework. Each health-check query triggers compute usage, and charges are incurred regardless of query purpose or origin. As a result, maintaining open connection pools can generate costs even when no actual user workloads are running.

Connection pooling maintains a pool of persistent connections between applications and the Amazon Redshift Serverless endpoint. To ensure these connections remain healthy and available, pooling mechanisms often send lightweight or empty queries (for example, `SELECT 1`) at regular intervals. These automated queries verify connection status.

When you use connection pooling, consider these best practices to minimize unintended charges:

- Adjust health check frequency by reviewing and optimizing the frequency of health check or keep-alive queries in your connection pooling configuration.
- Optimize idle system settings by configuring connection pooling to minimize unnecessary connection churn or background query activity during system idle times.
- Implement application-level pooling or improved connection lifecycle management if it can reduce overhead.
- Disable heartbeat or validation queries if your connection pooling configuration allows it. Check your specific connection string parameters or configuration files to adjust these settings.
- Fine-tune TCP keepalive settings: If you can't disable the driver's internal heartbeat mechanisms, adjust Transmission Control Protocol (TCP) keepalive settings at the operating system or application level to address connection timeout issues. Refer to your operating system, JDBC/ODBC driver, or connection pool documentation for details.

- Optimize database connection pooling: Configure your connection pool (HikariCP, Apache Database Connection Pool) to manage connections and minimize connection overhead. Focus on parameters such as maximum connections, idle timeout, and validation queries (if necessary). This optimization helps align Amazon Redshift Serverless compute usage with actual workload demand, potentially reducing costs.

Cost optimization for Amazon Redshift Serverless with zero-ETL

To optimize costs while running zero-ETL integrations on Amazon Redshift Serverless, you can right-size your environments and adjust your refresh settings based on workload needs. Consider making the following adjustments:

- Use the lower base RPU capacity of 8 RPU where available for workloads.
- Configure the REFRESH_INTERVAL of your target Redshift instance to balance freshness with cost. Shorter intervals ensure near real-time updates but drive up compute costs. Longer intervals (5 minutes or longer) reduce charges for workloads where immediate freshness is not critical, such as reporting or historical analysis. To edit your Redshift target REFRESH_INTERVAL, see the refresh interval clause in the ALTER DATABASE description.
- Maximize utilization of your Amazon Redshift Serverless environment by concurrently running analytics workloads while zero-ETL data is being ingested. This ensures that compute capacity is actively serving multiple business purposes.

Connecting to Amazon Redshift Serverless

Once you've set up your Amazon Redshift Serverless instance, you can connect to it in a variety of methods, outlined below. If you have multiple teams or projects and want to manage costs separately, you can use separate AWS accounts.

For a list of AWS Regions where the Amazon Redshift Serverless is available, see the endpoints listed for Redshift Serverless API in the *Amazon Web Services General Reference*.

Amazon Redshift Serverless connects to the serverless environment in your AWS account in the current AWS Region. Amazon Redshift Serverless runs in a VPC within the port ranges 5431-5455 and 8191-8215. The default is 5439. Currently, you can only change ports with the API operation `UpdateWorkgroup` and the AWS CLI operation `update-workgroup`.

Connecting to Amazon Redshift Serverless

You can connect to a database (named dev) in Amazon Redshift Serverless with the following syntax.

```
workgroup-name.account-number.aws-region.redshift-serverless.amazonaws.com:port/dev
```

For example, the following connection string specifies Region us-east-1.

```
default.123456789012.us-east-1.redshift-serverless.amazonaws.com:5439/dev
```

Connecting to Amazon Redshift Serverless through JDBC drivers

You can use one of the following methods to connect to Amazon Redshift Serverless with your preferred SQL client using the Amazon Redshift-provided JDBC driver version 2.x driver.

To connect with sign-in credentials for database authentication using the JDBC driver version 2.x, use the following syntax. The port number is optional; if not included, Amazon Redshift Serverless defaults to port number 5439. You can change to another port from the port range of 5431-5455 or 8191-8215. To change the default port for a serverless endpoint, use the AWS CLI and Amazon Redshift API.

```
jdbc:redshift://workgroup-name.account-number.aws-region.redshift-
serverless.amazonaws.com:5439/dev
```

For example, the following connection string specifies the workgroup default, the account ID 123456789012, and the Region us-east-2.

```
jdbc:redshift://default.123456789012.us-east-2.redshift-serverless.amazonaws.com:5439/
dev
```

To connect with IAM using the JDBC driver version 2.x, use the following syntax. The port number is optional; if not included, Amazon Redshift Serverless defaults to port number 5439. You can change to another port from the port range of 5431-5455 or 8191-8215. To change the default port for a serverless endpoint, use the AWS CLI and Amazon Redshift API.

```
jdbc:redshift:iam://workgroup-name.account-number.aws-region.redshift-
serverless.amazonaws.com:5439/dev
```

For example, the following connection string specifies the workgroup default, the account ID 123456789012, and the Region us-east-2.

```
jdbc:redshift:iam://default.123456789012.us-east-2.redshift-
serverless.amazonaws.com:5439/dev
```

For ODBC, use the following syntax.

```
Driver={Amazon Redshift (x64)}; Server=workgroup-name.account-number.aws-
region.redshift-serverless.amazonaws.com; Database=dev
```

If you are using a JDBC driver version prior to 2.1.0.9 and connecting with IAM, you will need to use the following syntax.

```
jdbc:redshift:iam://redshift-serverless-<name>:aws-region/database-name
```

For example, the following connection string specifies the workgroup default and the AWS Region us-east-1.

```
jdbc:redshift:iam://redshift-serverless-default:us-east-1/dev
```

For more information about drivers, see Configuring connections in Amazon Redshift.

Finding your JDBC and ODBC connection string

To connect to your workgroup with your SQL client tool, you must have the JDBC or ODBC connection string. You can find the connection string in the Amazon Redshift Serverless console, on a workgroup's details page.

To find the connection string for a workgroup

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Redshift Serverless**.
3. On the navigation menu, choose **Workgroup configuration**, then choose the workgroup name from the list to open its details.
4. The **JDBC URL** and **ODBC URL** connection strings are available, along with additional details, in the **General information** section. Each string is based on the AWS Region where the

workgroup runs. Choose the icon next to the appropriate connection string to copy the connection string.

Connecting to Amazon Redshift Serverless with the Data API

You can also use the Amazon Redshift Data API to connect to Amazon Redshift Serverless. Use the `workgroup-name` parameter instead of the `cluster-identifier` parameter in your AWS CLI calls.

For more information about the Data API, see Using the Amazon Redshift Data API. For example code calling the Data API in Python and other examples, see Getting Started with Redshift Data API and look in the `quick-start` and `use-cases` folders in *GitHub*.

Connecting with SSL to Amazon Redshift Serverless

Configuring a secure connection to Amazon Redshift Serverless

To support SSL connections, Redshift Serverless creates and installs an AWS Certificate Manager (ACM) issued SSL certificate for each workgroup. ACM certificates are publicly trusted by most operating systems, web browsers, and clients. You might need to download a certificate bundle if your SQL clients or applications connect to Redshift Serverless using SSL with the `sslmode` connection option set to `require`, `verify-ca`, or `verify-full`. If your client needs a certificate, Redshift Serverless provides a bundle certificate as follows:

- Download the bundle from https://s3.amazonaws.com/redshift-downloads/amazon-trust-ca-bundle.crt.
  - The expected MD5 checksum number is 418dea9b6d5d5de7a8f1ac42e164cdcf.
  - The sha256 checksum number is 36dba8e4b8041cd14b9d60158893963301bcbb92e1c456847784de2acb5bd550.

  Don't use the previous certificate bundle that was located at `https://s3.amazonaws.com/redshift-downloads/redshift-ca-bundle.crt`.
- In the China AWS Region, download the bundle from https://s3.cn-north-1.amazonaws.com.cn/redshift-downloads-cn/amazon-trust-ca-bundle.crt.
  - The expected MD5 checksum number is 418dea9b6d5d5de7a8f1ac42e164cdcf.
  - The sha256 checksum number is 36dba8e4b8041cd14b9d60158893963301bcbb92e1c456847784de2acb5bd550.

Don't use the previous certificate bundles that were located at `https://s3.cn-north-1.amazonaws.com.cn/redshift-downloads-cn/redshift-ca-bundle.crt` and `https://s3.cn-north-1.amazonaws.com.cn/redshift-downloads-cn/redshift-ssl-ca-cert.pem`

> **Important**
>
> Redshift Serverless has changed the way that SSL certificates are managed. You might need to update your current trust root CA certificates to continue to connect to your workgroups using SSL. For more information about ACM certificates for SSL connections, see Transitioning to ACM certificates for SSL connections.

By default, workgroup databases accept a connection whether it uses SSL or not.

To create a new workgroup that only accepts SSL connections, use the `create-workgroup` command and set the `require_ssl` parameter to `true`. To use the following example, replace *yourNamespaceName* with the name of your namespace and replace *yourWorkgroupName* with the name of your workgroup.

```
aws redshift-serverless create-workgroup \
--namespace-name yourNamespaceName \
--workgroup-name yourWorkgroupName \
--config-parameters parameterKey=require_ssl,parameterValue=true
```

To update an existing workgroup to only accept SSL connections, use the `update-workgroup` command and set the `require_ssl` parameter to `true`. Note that Redshift Serverless will restart your workgroup when you update the `require_ssl` parameter. To use the following example, replace *yourWorkgroupName* with the name of your workgroup.

```
aws redshift-serverless update-workgroup \
--workgroup-name yourWorkgroupName \
--config-parameters parameterKey=require_ssl,parameterValue=true
```

Amazon Redshift supports the Elliptic Curve Diffie—Hellman Ephemeral (ECDHE) key agreement protocol. With ECDHE, the client and server each have an elliptic curve public-private key pair that is used to establish a shared secret over an insecure channel. You don't need to configure anything

in Amazon Redshift to enable ECDHE. If you connect from a SQL client tool that uses ECDHE to encrypt communication between the client and server, Amazon Redshift uses the provided cipher list to make the appropriate connection. For more information, see Elliptic curve diffie—hellman on Wikipedia and Ciphers on the OpenSSL website.

Configuring a FIPS-compliant SSL connection to Amazon Redshift Serverless

To create a new workgroup that uses a FIPS-compliant SSL connection, use the `create-workgroup` command and set the `use_fips_ssl` and `require_ssl` parameters to `true`. To use the following example, replace *`yourNamespaceName`* with the name of your namespace and replace *`yourWorkgroupName`* with the name of your workgroup.

```
aws redshift-serverless create-workgroup \
--namespace-name yourNamespaceName \
--workgroup-name yourWorkgroupName \
--config-parameters '[{"parameterKey": "require_ssl", "parameterValue": "true"},
 {"parameterKey": "use_fips_ssl", "parameterValue": "true"}]'
```

To update an existing workgroup to use a FIPS-compliant SSL connection, use the `update-workgroup` command and set the `use_fips_ssl` and `require_ssl` parameters to `true`. Note that Redshift Serverless will restart your workgroup when you update the `use_fips_ssl` parameter. To use the following example, replace *`yourWorkgroupName`* with the name of your workgroup.

```
aws redshift-serverless update-workgroup \
--workgroup-name yourWorkgroupName \
--config-parameters '[{"parameterKey": "require_ssl", "parameterValue": "true"},
 {"parameterKey": "use_fips_ssl", "parameterValue": "true"}]'
```

For more information about configuring Redshift Serverless to use FIPS-compliant connections, see use_fips_ssl in the *Amazon Redshift Database Developer Guide*.

Connecting to Amazon Redshift Serverless from an Amazon Redshift managed VPC endpoint

Connecting to Amazon Redshift Serverless from other VPC endpoints

For information about setting up or configuring a managed-VPC endpoint for an Amazon Redshift Serverless workgroup, see Working with Redshift-managed VPC endpoints.

Connecting to Amazon Redshift Serverless from an interface VPC endpoint (AWS PrivateLink)

For information about connecting to Amazon Redshift Serverless from an interface VPC endpoint (AWS PrivateLink), see Interface VPC endpoints.

Connecting to Amazon Redshift Serverless from a Redshift VPC endpoint in another account

Connecting to Amazon Redshift Serverless from a cross VPC endpoint

Amazon Redshift Serverless is provisioned in a VPC. You can grant access to a VPC in another account to access Amazon Redshift Serverless in your account. This is similar to a connection from a managed VPC endpoint, but in this case the connection originates, for example, from a database client in another account. There are a few operations that you can perform:

- A database owner can grant access to a VPC containing Amazon Redshift Serverless to another account in the same Region.
- A database owner can revoke Amazon Redshift Serverless access.

The main benefit of cross-account access is allowing easier database collaboration. Users don't have to be provisioned in the account that contains the database to access it, which reduces configuration steps and saves time.

Permissions required to grant access to a VPC in another account

To grant access or change the access allowed, the grantor requires an assigned permissions policy with the following permissions:

- redshift-serverless:PutResourcePolicy
- redshift-serverless:GetResourcePolicy
- redshift-serverless:DeleteResourcePolicy
- ec2:CreateVpcEndpoint
- ec2:ModifyVpcEndpoint

You might need other permissions that are specified in the AWS managed policy *AmazonRedshiftFullAccess*. For more information, see Granting permissions to Amazon Redshift Serverless.

The grantee requires an assigned permissions policy with the following permissions:

- redshift-serverless:ListWorkgroups
- redshift-serverless:CreateEndpointAccess
- redshift-serverless:UpdateEndpointAccess
- redshift-serverless:GetEndpointAccess
- redshift-serverless:ListEndpointAccess
- redshift-serverless:DeleteEndpointAccess

As a best practice, we recommend attaching permissions policies to an IAM role and then assigning it to users and groups as needed. For more information, see Identity and access management in Amazon Redshift.

This is a sample resource policy used to configure cross-VPC access:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CrossAccountCrossVPCAccess",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "123456789012",
                    "234567890123"
                ]
            },
            "Action": [
                "redshift-serverless:CreateEndpointAccess",
                "redshift-serverless:UpdateEndpointAccess",
                "redshift-serverless:DeleteEndpointAccess",
                "redshift-serverless:GetEndpointAccess"
            ],
```

```
            "Resource": "arn:aws:redshift-serverless:*:*:*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": "us-east-1"
                },
                "StringLike": {
                    "aws:SourceVpc": [
                       "vpc-11223344556677889",
                       "vpc-12345678"
                    ]
                }
            }
        }
    ]
}
```

The procedures that follow in this section assume that the user performing them has the appropriate assigned permissions, for example, by means of an assigned IAM role that has the permissions listed. The procedures also assume that the workgroup has an IAM role attached with appropriate resource permissions.

Granting VPC access to other accounts, using the console

This procedure shows the steps for configuring database access when you're the database owner, and you want to grant access to it.

Granting access from the owner account

1. In the properties for the Amazon Redshift Serverless workgroup, on the **Data access** tab, there is a list called **Granted accounts**. It shows accounts and VPCs granted access to the workgroup. Find the list and choose **Grant access** to add an account to the list.
2. A window appears where you can add the grantee information. Enter the AWS account ID, which is the 12-digit ID of the account that you want to grant access to.
3. Grant access to all VPCs for the grantee, or to specific VPCs. If you grant access only to specific VPCs, you can add the IDs for these by entering each one and choosing **Add VPC**.
4. **Save changes** when you're finished.

When you save the changes, the account appears in the list of **Granted accounts**. The entry shows the **Account ID** and the list of VPCs granted access.

The database owner can also revoke access to an account. The owner can revoke access at any time.

Revoking access to an account

1. You can start from the list of granted accounts. First, select one or more accounts.
2. Choose **Revoke access**.

After access is granted, a database administrator for the grantee can check the console to determine if they have access.

Using the console to confirm that access is granted for you to access another account

1. In the Amazon Redshift Serverless workgroup properties, on the **Data access** tab, there is a list called **Authorized accounts**. It shows accounts that can be accessed from this workgroup. The grantee can't use the workgroup's endpoint URL to access to the workgroup directly. To access the workgroup, you as the grantee go to the **endpoint** section and choose **create an endpoint**.
2. Then, as the grantee, you provide an endpoint name and a VPC to access the workgroup.
3. After the endpoint is created successfully, it appears in the **endpoint** section and there is an endpoint URL for it. You can use this endpoint URL to access the workgroup.

Granting access to other accounts, using CLI commands

The account granting access must first grant access to another account to connect by using `put-resource-policy`. The database owner can call `put-resource-policy` to authorize another account to create connections to the workgroup. The grantee account can then use `create-endpoint-authorization` to create connections to the workgroup through their allowed VPCs.

The following shows the properties for `put-resource-policy`, which you can call to allow access to a specific account and VPC.

```
aws redshift-serverless put-resource-policy
--resource-arn <value>
--policy <value>
```

After calling the command, you can call `get-resource-policy`, specifying the `resource-arn` to see which accounts and VPCs are allowed to access the resource.

The following call can be made by the grantee. It shows information about the granted access. Specifically, it returns a list that contains the VPCs granted access.

```
aws redshift-serverless list-workgroups
--owner-account <value>
```

The purpose of this is for the grantee to get information from the granting account about endpoint authorizations. The `owner-account` is the sharing account. When you run this, it returns the `CrossAccountVpcs` for each workgroup, which is a list of allowed VPCs. For reference, the following shows all of the properties available for a workgroup:

```
Output: workgroup (Object)
workgroupId String,
workgroupArn String,
workgroupName String,
status: String,
namespaceName: String,
baseCapacity: Integer, (Not-applicable)
enhancedVpcRouting: Boolean,
configParameters: List,
securityGroupIds: List,
subnetIds: List,
endpoint: String,
publiclyAccessible: Boolean,
creationDate: Timestamp,
port: Integer,
CrossAccountVpcs: List
```

Note

As a point of reminder, cluster relocation isn't a prerequisite for configuring additional Redshift networking features. It also isn't required that you turn it on to enable the following:

- **Connecting from a cross-account or cross-region VPC to Redshift** – You can connect from one AWS virtual private cloud (VPC) to another that contains a Redshift database, as described in this section.
- **Setting up a custom domain name** – You can create a custom domain name, also known as a custom URL, for your Amazon Redshift cluster or Amazon Redshift Serverless

workgroup, to make the endpoint name more memorable and simple. For more information, see Using a custom domain name for client connections.

Additional resources

Instructions for setting your network traffic settings are available in Public accessibility with default or custom security group configuration. This includes a use case where the cluster is publicly accessible.

Instructions for setting your network traffic settings are available in Private accessibility with default or custom security group configuration. This includes a use case where the cluster isn't available to the internet.

For more information about secure connections to Amazon Redshift Serverless, including granting permissions, authorizing access to additional services, and creating IAM roles, see Identity and access management in Amazon Redshift Serverless.

Defining database roles to grant to federated users in Amazon Redshift Serverless

When you're part of an organization, you have a collection of associated roles. For instance, you have roles for your job function, like *programmer* and *manager*. Your roles determine which applications and data you have access to. Most organizations use an identity provider, such as Microsoft Active Directory, to assign roles to users and groups. The use of roles to control resource access has grown, because organizations don't have to do as much management of individual users.

Recently, role-based access control was introduced in Amazon Redshift Serverless. Using database roles, you can secure access to data and objects, like schemas or tables, for example. Or you can use roles to define a set of elevated permissions, such as for a system monitor or database administrator. But after you grant resource permissions to database roles, there is an additional step, which is to connect a user's roles from the organization to the database roles. You can assign each user to their database roles upon initial sign in by running SQL statements, but it's a lot of effort. An easier way is to define the database roles to grant and pass them to Amazon Redshift Serverless. This has the advantage of simplifying the initial sign-in process.

You can pass roles to Amazon Redshift Serverless using `GetCredentials`. When a user signs in for the first time to an Amazon Redshift Serverless database, an associated database user is created and mapped to the matching database roles. This topic details the mechanism for passing roles to Amazon Redshift Serverless.

Passing database roles has a couple primary use cases:

- When a user signs in through a third-party identity provider, typically with federation configured, and passes the roles by means of a session tag.
- When a user signs in through IAM sign-in credentials, and their roles are passed by means of a tag key and value.

For more information about role-based access control, see Role-based access control (RBAC).

Defining database roles

Before you can pass roles to Amazon Redshift Serverless, you must configure database roles in your database and grant them appropriate permissions on database resources. For instance, in a simple scenario, you can create a database role named *sales* and grant it access to query tables with sales data. For more information about how to create database roles and grant permissions, see CREATE ROLE and GRANT.

Use cases for defining database roles to grant to federated users

These sections outline a couple use cases where passing database roles to Amazon Redshift Serverless can simplify access to database resources.

Signing in using an identity provider

The first use case assumes that your organization has user identities in an identity and access management service. This service can be cloud based, for example JumpCloud or Okta, or on-premises, such as Microsoft Active Directory. The goal is to automatically map a user's roles from the identity provider to your database roles when they sign in to a client like Query editor V2, for instance, or with a JDBC client. To set this up, you must complete a couple of configuration tasks. These include the following:

1. Configure federated integration with your identity provider (IdP) using a trust relationship. This is a prerequisite. When you set this up, the identity provider is responsible for authenticating

the user via a SAML assertion and providing sign-in credentials. For more information, see Integrating third party SAML solution providers with AWS. You can also find more information at Federate access to Amazon Redshift query editor V2 with Active Directory Federation Services (AD FS) or Federate single sign-on access to Amazon Redshift query editor v2 with Okta.

2. The user must have the following policy permissions:

   - `GetCredentials` – Provides credentials for temporary authorization to log in to Amazon Redshift Serverless.
   - `sts:AssumeRoleWithSAML` – Provides a mechanism for tying an enterprise identity store or directory to role-based AWS access.
   - `sts:TagSession` – Permission to the tag-session action, on the identity provider principal.

   In this case, `AssumeRoleWithSAML` returns a set of security credentials for users who have been authenticated via a SAML authenticated response. This operation provides a mechanism for tying an identity store or directory to role-based AWS access without user-specific credentials. For users with permission to `AssumeRoleWithSAML`, the identity provider is responsible for managing the SAML assertion that is used to pass the role information.

   As a best practice, we recommend attaching permissions policies to an IAM role and then assigning it to users and groups as needed. For more information, see Identity and access management in Amazon Redshift.

3. You configure the tag `RedshiftDbRoles` with the colon-separated role values, in the format *role1:role2*. For example, `manager:engineer`. These can be retrieved from a session-tag implementation configured in your identity provider. The SAML authentication request passes the roles programmatically. For more information about passing session tags, see Passing session tags in AWS STS.

   In a case where you pass a role name that doesn't exist in the database, it's ignored.

In this use case, when a user signs in using a federated identity, their roles are passed in the authorization request through the session tag key and value. Subsequently, following authorization, `GetCredentials` passes the roles to the database. Upon a successful connection, the database roles are mapped and the user can perform database tasks corresponding with their role. The essential part of the operation is that the `RedshiftDbRoles` session tag is assigned the roles in the initial authorization request. For more information about passing session tags, see Passing session tags using AssumeRoleWithSAML.

Signing in using IAM credentials

In the second use case, roles can be passed for a user and they can access a database client application through IAM credentials.

1. The user who signs in in this case must be assigned policy permissions for the following actions:
   - `tag:GetResources` – Returns tagged resources associated with specified tags.
   - `tag:GetTagKeys` – Returns tag keys currently in use.

     As a best practice, we recommend attaching permissions policies to an IAM role and then assigning it to users and groups as needed. For more information, see Identity and access management in Amazon Redshift.
2. Allow permissions are also required to access the database service, such as Amazon Redshift Serverless.
3. For this use case, configure the tag values for your roles in AWS Identity and Access Management. You can choose to **edit tags** and create a tag key called *RedshiftDbRoles* with an accompanying tag value string that contains the roles. For example, *manager:engineer*.

When a user logs in, their role is added to the authorization request and passed to the database. It is mapped to an existing database role.

Additional resources

As mentioned in the use cases, you can configure the trust relationship between your IdP and AWS. For more information, see Configuring your SAML 2.0 IdP with relying party trust and adding claims.

Identity and access management in Amazon Redshift Serverless

Access to Amazon Redshift requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access AWS resources, such as Amazon Redshift Serverless.

The following sections provide details about how you can use AWS Identity and Access Management (IAM) and Amazon Redshift to help secure your resources by controlling who can access them. For more information, see Identity and access management in Amazon Redshift.

Granting permissions to Amazon Redshift Serverless

To access other AWS services, Amazon Redshift Serverless requires permissions. Some Amazon Redshift features require Amazon Redshift to access other AWS services on your behalf. For your Amazon Redshift Serverless instance to act for you, supply security credentials to it. The preferred method to supply security credentials is to specify an AWS Identity and Access Management (IAM) role. You can also create an IAM role through the Amazon Redshift console and set it as the default. For more information, see Creating an IAM role as default for Amazon Redshift.

To access other AWS services, create an IAM role with the appropriate permissions. You also need to associate the role with Amazon Redshift Serverless. In addition, either specify the Amazon Resource Name (ARN) of the role when you run the Amazon Redshift command or specify the `default` keyword.

When changing the trust relationship for the IAM role in the https://console.aws.amazon.com/iam/, make sure that it contains `redshift-serverless.amazonaws.com` and `redshift.amazonaws.com` as principal service names. For information about how to manage IAM roles to access other AWS services on your behalf, see Authorizing Amazon Redshift to access AWS services on your behalf.

Creating an IAM role as default for Amazon Redshift

When you create IAM roles through the Amazon Redshift console, Amazon Redshift programmatically creates the roles in your AWS account. Amazon Redshift also automatically attaches existing AWS managed policies to them. This approach means that you can stay within the Amazon Redshift console and don't have to switch to the IAM console for role creation.

The IAM role that you create through the console for your cluster has the `AmazonRedshiftAllCommandsFullAccess` managed policy automatically attached. This IAM role allows Amazon Redshift to copy, unload, query, and analyze data for AWS resources in your IAM account. The related commands include COPY, UNLOAD, CREATE EXTERNAL FUNCTION, CREATE EXTERNAL TABLE, CREATE EXTERNAL SCHEMA, CREATE MODEL, and CREATE LIBRARY. For more information about how to create an IAM role as default for Amazon Redshift, see Creating an IAM role as default for Amazon Redshift.

To get started creating an IAM role as default for Amazon Redshift, open the AWS Management Console, choose the Amazon Redshift console, and then choose **Redshift Serverless** in the menu. From the Serverless dashboard you can create a new workgroup. The creation steps walk you selecting an IAM role or configuring a new IAM one.

When you have an existing Amazon Redshift Serverless workgroup and you want to configure IAM roles for it, open the AWS Management Console. Choose the Amazon Redshift console, and then choose **Redshift Serverless**. On the Amazon Redshift Serverless console, choose **Namespace configuration** for an existing workgroup. Under **Security and encryption**, you can edit the permissions.

Assigning IAM roles to a namespace

Each IAM role is an AWS identity with permissions policies that determine what actions each role can perform in AWS. The role is intended to be assumable by anyone who needs it. Additionally, each namespace is a collection of objects, like tables and schemas, and users. When you use Amazon Redshift Serverless, you can associate multiple IAM roles with your namespace. This makes it easier to structure your permissions appropriately for a collection of database objects, so roles can perform actions on both internal and external data. For example, so you can run a `COPY` command in an Amazon Redshift database to retrieve data from Amazon S3 and populate a Redshift table.

You can associate multiple roles to a namespace using the console, as described previously in this section. You can also use the API command `CreateNamespace`, or the CLI command `create-namespace`. With the API or CLI command, you can assign IAM roles to the namespace by populating `IAMRoles` with one or more roles. Specifically, you add ARNs for specific roles to the collection.

Managing namespace associated IAM roles

On the AWS Management Console you can manage permissions policies for roles in AWS Identity and Access Management. You can manage IAM roles for the namespace, using settings available under **Namespace configuration**. For more information about namespaces and their use in Amazon Redshift Serverless, see Workgroups and namespaces.

Getting started with IAM credentials for Amazon Redshift

When you sign in to the Amazon Redshift console for the first time and first try out Amazon Redshift Serverless, we recommend that you sign in as a user with an attached IAM role that has the policies required. After you get started creating an Amazon Redshift Serverless instance, Amazon Redshift records the IAM role name that you used to sign in. You can use the same credentials to sign in to the Amazon Redshift console and the Amazon Redshift Serverless console.

While creating the Amazon Redshift Serverless instance, you can create a database. Use the query editor v2 to connect to the database with the temporary credentials option.

To add a new admin user name and password that persist for the database, choose **Customize admin user credentials** and enter a new admin user name and admin user password.

To get started using Amazon Redshift Serverless and create a workgroup and namespace in the console for the first time, use an IAM role with a permissions policy attached. Make sure that this role has either the administrator permission `arn:aws:iam::aws:policy/AdministratorAccess` or the full Amazon Redshift permission `arn:aws:iam::aws:policy/AmazonRedshiftFullAccess` attached to the IAM policy.

The following scenarios outline how your IAM credentials are used by Amazon Redshift Serverless when you get started on the Amazon Redshift Serverless console:

- If you choose **Use default settings** – Amazon Redshift Serverless translates your current IAM identity to a database superuser. You can use the same IAM identity with the Amazon Redshift Serverless console to perform superuser actions in your database in Amazon Redshift Serverless.
- If you choose **Customize settings** without specifying the **Admin user name** and password Amazon Redshift Serverless, your current IAM credentials are used as your default admin user credentials.
- If you choose **Customize settings** and specify **Admin user name** and password Amazon Redshift Serverless – Amazon Redshift Serverless translates your current IAM identity to a database superuser. Amazon Redshift Serverless also creates another long-term login username and password pair also as a superuser. You can either use your current IAM identity or the created username and password pair to login in to your database as a superuser.

Accessing Amazon Redshift Serverless database objects with database-role permissions

This procedure shows how to grant permission to query a table through an Amazon Redshift database role. The role is assigned by means of a tag that's attached to a user in IAM and passed to Amazon Redshift when they sign in. It's an explanation by example of the concepts in Defining database roles to grant to federated users in Amazon Redshift Serverless. The benefit of completing these steps is that you can associate a user with a database role and avoid setting their permissions for each database object. It simplifies managing the user's ability to query, modify, or add data to tables and to perform other actions.

The procedure assumes you have already set up an Amazon Redshift Serverless database and you have the ability to grant permissions in the database. It also assumes you have permissions to create an IAM user in the AWS console, to create an IAM role, and to assign policy permissions.

1. Create an IAM user, using the IAM console. Later, you will connect to the database with this user.
2. Create a Redshift database role, using query editor v2 or another SQL client. For more information on creating database roles, see CREATE ROLE.

   ```
   CREATE ROLE urban_planning;
   ```

   Query the SVV_ROLES system view to check that your role is created. It also returns system roles.

   ```
   SELECT * from SVV_ROLES;
   ```
3. Grant the database role you created permission to select from a table. (The IAM user you created will eventually sign in and select records from the table by means of the database role.) The role name and table name in the following code example are samples. Here, permission is granted to select from a table named `cities`.

   ```
   GRANT SELECT on TABLE cities to ROLE urban_planning;
   ```
4. Use the AWS Identity and Access Management console to create an IAM role. This role grants permission to use query editor v2. Create a new IAM role and, for the trusted entity type, choose **AWS account**. Then choose **This account**. Give the role the following policy permissions:

   - `AmazonRedshiftReadOnlyAccess`
   - `tag:GetResources`
   - `tag:GetTagKeys`
   - All actions for sqlworkbench, including `sqlworkbench:ListDatabases` and `sqlworkbench:UpdateConnection`.
5. In the IAM console, add a tag with the **Key** `RedshiftDbRoles` to the IAM user you created previously. The tag's value should match the database role you created in the first step. It's `urban_planning` in the sample.

After you complete these steps, assign the IAM role to the user you created in the IAM console. When the user signs in to the database with query editor v2, their database role name in the tag is passed to Amazon Redshift and associated with them. Thus, they can query the appropriate tables by means of the database role. To illustrate, the user in this sample can query the `cities` table through the `urban_planning` database role.

Migrating a provisioned cluster to Amazon Redshift Serverless

You can migrate your existing provisioned clusters to Amazon Redshift Serverless, enabling on-demand and automatic scaling of compute resources. Migrating a provisioned cluster to Amazon Redshift Serverless allows you to optimize costs by paying only for the resources you use and automatically scaling capacity based on workload demands. Common use cases for the migration include running ad-hoc queries, periodic data processing jobs, or handling unpredictable workloads without over-provisioning resources. Perform the following set of tasks to migrate your provisioned Amazon Redshift cluster to the serverless deployment option.

Creating a snapshot of your provisioned cluster

> **Note**
>
> Amazon Redshift automatically converts interleaved keys to compound keys when you restore a provisioned cluster snapshot to a serverless namespace.

To transfer data from your provisioned cluster to Amazon Redshift Serverless, create a snapshot of your provisioned cluster, and then restore the snapshot in Amazon Redshift Serverless.

> **Note**
>
> Before you migrate your data to a serverless workgroup, ensure that your provisioned cluster needs are compatible with the amount of RPU you choose in Amazon Redshift Serverless.

To create a snapshot of your provisioned cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.

2. On the navigation menu, choose **Clusters**, **Snapshots**, and then choose **Create snapshot**.
3. Enter the properties of the snapshot definition, then choose **Create snapshot**. It might take some time for the snapshot to be available.

To restore a provisioned cluster snapshot to a serverless namespace:

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. Start on the Amazon Redshift provisioned cluster console and navigate to the **Clusters**, **Snapshots** page.
3. Choose a snapshot to use.
4. Choose **Restore snapshot**, **Restore to serverless namespace**.
5. Choose a namespace to restore your snapshot to.
6. Confirm you want to restore from your snapshot. This action replaces all the databases in your serverless endpoint with the data from your provisioned cluster. Choose **Restore**.

For more information about provisioned cluster snapshots, see Amazon Redshift snapshots.

Connecting to Amazon Redshift Serverless using a driver

To connect to Amazon Redshift Serverless with your preferred SQL client, you can use the Amazon Redshift provided JDBC driver version 2.x. We recommend connecting to Amazon Redshift using the latest version of the Amazon Redshift JDBC driver version 2.x. The port number is optional. If you don't include it, Amazon Redshift Serverless defaults to port number 5439. You can change to another port from the port range of 5431-5455 or 8191-8215. To change the default port for a serverless endpoint, use the AWS CLI and Amazon Redshift API.

To find the exact endpoint to use for the JDBC, ODBC, or Python driver, see **Workgroup configuration** in Amazon Redshift Serverless. You can also use the Amazon Redshift Serverless API operation `GetWorkgroup` or the AWS CLI operation `get-workgroups` to return information about your workgroup, and then connect.

Connecting using password-based authentication

To establish a connection using the Amazon Redshift JDBC driver version 2.x with password-based authentication, use the following syntax:

```
jdbc:redshift://<workgroup-name>.<account-number>.<aws-region>.redshift-
serverless.amazonaws.com:5439/?username=username&password=password
```

To establish a connection using the Amazon Redshift Python connector with password-based authentication, use the following syntax:

```
import redshift_connector
with redshift_connector.connect(
    host='<workgroup-name>.<account-number>.<aws-region>.redshift-
serverless.amazonaws.com',
    database='<database-name>',
    user='username',
    password='password'
    # port value of 5439 is specified by default
) as conn:
    pass
```

To establish a connection using the Amazon Redshift ODBC driver version 2.x with password-based authentication, use the following syntax:

```
Driver={Amazon Redshift ODBC Driver (x64)}; Server=<workgroup-name>.<account-
number>.<aws-region>.redshift-serverless.amazonaws.com; Database=database-name;
 User=username; Password=password
```

Connecting using IAM

If you prefer logging in with IAM, use the Amazon Redshift Serverless GetCredentials API operation.

To use IAM authentication, add `iam:` to the Amazon Redshift JDBC URL following `jdbc:redshift:`, as shown in the following example.

```
jdbc:redshift:iam://<workgroup-name>.<account-number>.<aws-region>.redshift-
serverless.amazonaws.com:5439/<database-name>
```

This Amazon Redshift Serverless endpoint doesn't support customizing dbUser, dbGroup, or auto-create. By default, the driver automatically creates database users at login. It then assigns the users to Amazon Redshift database roles based on the tags specified in IAM, or based on the groups defined in your identity provider (IdP).

Ensure that your AWS identity has the correct IAM policy for the `redshift-serverless:GetCredentials` action. The following is an example IAM policy that grants the correct permissions to an AWS identity to connect to Amazon Redshift Serverless. For more information about IAM permissions, see Adding and removing IAM identity permissions in the *IAM User Guide*.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Action": "redshift-serverless:GetCredentials",
            "Resource": "*"
        }
    ]
}
```

To establish a connection using the Amazon Redshift Python connector with IAM based authentication, use `iam=true` in your code, as shown in the following syntax:

```
import redshift_connector
with redshift_connector.connect(
    iam=True,
    host='<workgroup-name>.<account-number>.<aws-region>.redshift-
serverless.amazonaws.com',
    database='<database-name>'
    <IAM credentials>
) as conn:
    pass
```

For `IAM credentials`, you can use any credentials, including the following:

- AWS profile configuration.
- IAM credentials (an access key ID, secret access key, and optionally a session token).
- Identity provider federation.

To establish a connection using the Amazon Redshift ODBC driver version 2.x with IAM based authentication and a profile, use the following syntax:

```
Driver={Amazon Redshift ODBC Driver (x64)}; IAM=true; Server=<workgroup-name>.<account-number>.<aws-region>.redshift-serverless.amazonaws.com; Database=database-name;
 Profile=aws-profile-name;
```

Connecting using IAM with the GetClusterCredentials API

> **Note**
>
> When connecting to Amazon Redshift Serverless, we recommend that you use the `GetCredentials` API. This API offers comprehensive role-based access control (RBAC) functionality as well as other new features that aren't available in `GetClusterCredentials`. We support the `GetClusterCredentials` API to simplify the transition from provisioned clusters to serverless workgroups, but we strongly recommend migrating to using `GetCredentials` as soon as possible for optimal compatibility.

You can establish a connection to Amazon Redshift Serverless using the `GetClusterCredentials` API. To implement this authentication method, modify your client or application by incorporating the following parameters:

- `iam=true`
- `clusterid/cluster_identifier=redshift-serverless-<workgroup-name>`
- `region=<aws-region>`

The following examples demonstrate the BrowserSAML plugin across all three drivers. This represents one of several available authentication approaches. The examples can be modified to use alternative authentication methods or plugins according to your specific requirements.

IAM policy permissions for GetClusterCredentials

Following is a sample IAM policy with the permissions required to use `GetClusterCredentials` with Amazon Redshift Serverless:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "redshift-serverless:GetWorkgroup",
                "redshift-serverless:GetNamespace",
                "redshift:GetClusterCredentials",
                "redshift:CreateClusterUser",
                "redshift:JoinGroup",
                "redshift:ExecuteQuery",
                "redshift:FetchResults",
                "redshift:DescribeClusters",
                "redshift:DescribeTable"
            ],
            "Resource": [
                "arn:aws:redshift-serverless:us-
east-1:111122223333:workgroup/workgroup-name",
                "arn:aws:redshift-serverless:us-
east-1:111122223333:namespace/namespace-name",
                "arn:aws:redshift:us-east-1:111122223333:cluster:cluster-name",
                "arn:aws:redshift:us-east-1:111122223333:dbuser:database-name/
${redshift:DbUser}"
            ]
        }
    ]
}
```

To establish a connection using the Amazon Redshift JDBC driver version 2.x with `GetClusterCredentials`, use the following syntax:

```
jdbc:redshift:iam://redshift-serverless-<workgroup-name>:<aws-region>/<database-name>?
plugin_name=com.amazon.redshift.plugin.BrowserSamlCredentialsProvider&login_url=<single
 sign-on URL from IdP>"
```

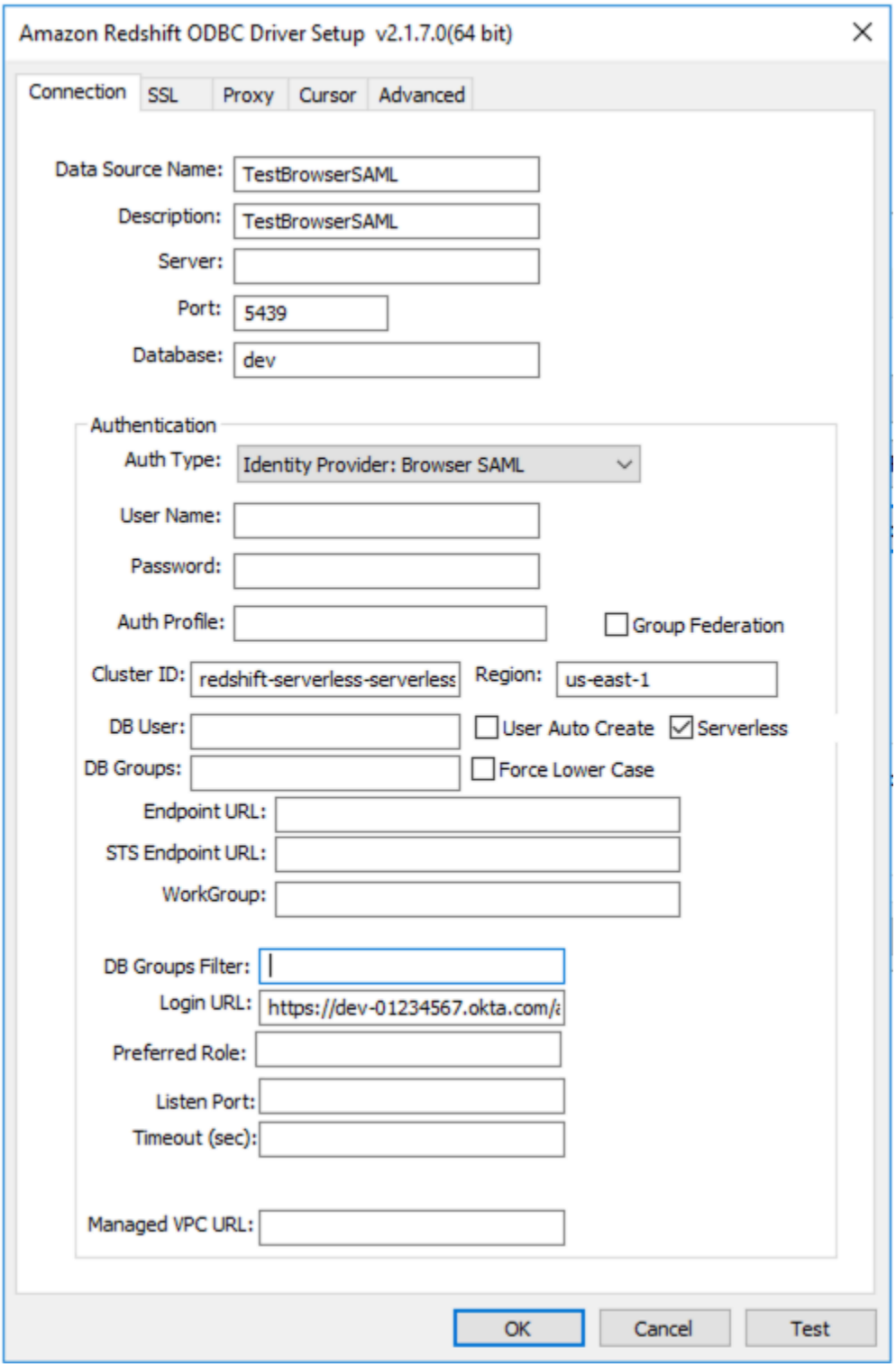
To establish a connection using the Amazon Redshift Python connector with `GetClusterCredentials`, use the following syntax:

```
import redshift_connector
with redshift_connector.connect(
    iam=True,
    cluster_identifier='redshift-serverless-<workgroup-name>',
    region='<aws-region>',
    database='<database-name>',
    credentials_provider='BrowserSamlCredentialsProvider'
    login_url='<single sign-on URL from IdP>'
    # port value of 5439 is specified by default
) as conn:
    pass
```

To establish a connection using the Amazon Redshift ODBC driver version 2.x with `GetClusterCredentials`, use the following syntax:

```
Driver= {Amazon Redshift ODBC Driver (x64)}; IAM=true; isServerless=true;
 ClusterId=redshift-serverless-<workgroup-name>; region=<aws-region>;
 plugin_name=BrowserSAML;login_url=<single sign-on URL from IdP>
```

Following is an example ODBC DSN configuration in Windows:



Using the Amazon Redshift Serverless SDK

If you wrote any management scripts using the Amazon Redshift SDK, you must use the new Amazon Redshift Serverless SDK to manage Amazon Redshift Serverless and associated resources.

For more information about available API operations, see the Amazon Redshift Serverless API Reference guide.

Workgroups and namespaces

To isolate workloads and manage different resources in Amazon Redshift Serverless, you can create namespaces and workgroups and manage storage and compute resources separately.

A namespace is a collection of database objects and users. The storage-related namespace groups together schemas, tables, users, or AWS Key Management Service keys for encrypting data. Storage properties include the database name and password of the admin user, permissions, and encryption and security. Other resources that are grouped under namespaces include datashares, recovery points, and usage limits. You can configure these storage properties using the Amazon Redshift Serverless console, the AWS Command Line Interface, or the Amazon Redshift Serverless APIs for the specific resource.

Workgroup is a collection of compute resources. The compute-related workgroup groups together compute resources like RPUs, VPC subnet groups, and security groups. Properties for the workgroup include network and security settings. Other resources that are grouped under workgroups include access and usage limits. You can configure these compute properties using the Amazon Redshift Serverless console, the AWS Command Line Interface, or the Amazon Redshift Serverless APIs.

You can create one or more namespaces and workgroups. Each namespace can have only one workgroup associated with it. Conversely, each workgroup can be associated with only one namespace.

Workgroups and namespaces using the console

Setting up Amazon Redshift Serverless involves walking through several configuration steps. When you follow the steps to set up Amazon Redshift Serverless, you create a namespace and workgroup, and associate them with each other. To get started setting Amazon Redshift Serverless configuration using the Amazon Redshift Serverless console, you can choose **Get started with Amazon Redshift Serverless** to set up Amazon Redshift Serverless and begin to interact with it. You can choose an environment with default settings, which makes for quicker setup, or explicitly configure the settings per your organization's requirements.During this process, you specify settings for your workgroup and namespace.

After you set up the environment, Workgroup properties and Namespace properties help you get familiar with the settings.

Workgroups and namespaces using the AWS Command Line Interface and Amazon Redshift Serverless API

Aside from using the AWS console, you can also use the AWS CLI or the Amazon Redshift Serverless API to interact with workgroups and namespaces. The table below lists the API and CLI operations you can use to manage snapshots and recovery points.

API operation	CLI command	Description
CreateNamespace	create-namespace	Creates a namespace. By default, Amazon Redshift Serverless creates namespaces with a default AWS Key Management Service key, but you can specify another key to encrypt your data. You can also create a namespace by restoring a snapshot. See Working with snapshots and recovery points for more information.
UpdateNamespace	update-namespace	Updates a namespace.
GetNamespace	get-namespace	Retrieves information about a namespace
ListNamepaces	list-namespaces	Retrieves information about a list of namespaces.
DeleteNamespace	delete-namespace	Deletes a namespace.
CreateWorkgroup	create-workgroup	Creates a workgroup. When creating a workgroup, make sure that you have an existing

API operation	CLI command	Description
		namespace that you can associate with the workgroup. When creating the workgroup , you can specify compute resources such as subnets, security groups, and RPUs.
UpdateWorkgroup	update-workgroup	Updates a workgroup.
GetWorkgroup	get-workgroup	Retrieves information about a workgroup.
ListWorkgroups	list-workgroups	Retrieves information about a list of workgroups.
DeleteWorkgroup	delete-workgroup	Deletes a workgroup.

Workgroups

With Amazon Redshift Serverless, you can create and manage workgroups to isolate and control compute resources for different workloads or users. Workgroups allow you to set configuration options like memory and concurrency scaling limits, and prioritize query execution across workloads. The compute-related workgroup groups together compute resources like RPUs and VPC subnet groups.

Creating a workgroup with a namespace

Complete the following steps to create a workgroup. For more information about workgroup configuration, see Workgroup properties.

1. Choose the **Serverless dashboard**. Then choose **Create workgroup**.
2. Enter the workgroup name.
3. Choose an **IP address type** for the workgroup. Choices include:

   - **IPv4** – With this option, your AWS resources only communicate over the IPv4 addressing protocol.

- **Dual-stack mode** – With this option, your AWS resources can communicate over the IPv4, IPv6, or both addressing protocols. Also, you must associate an IPv6 CIDR block with the VPC and subnets used for your workgroup in the Amazon VPC. You can use the Amazon VPC console to create an Amazon VPC or update an existing Amazon VPC to use IPv6 addressing. For more information, see IPv6 support for your VPC; in the *Amazon VPC User Guide*.

4. Choose a **Virtual private cloud (VPC)** for Amazon Redshift Serverless. This assigns the workgroup to a specific virtual network in your AWS environment. When using **dual-stack mode**, the Amazon VPC you choose must support IPV6 addressing. For more information about an Amazon VPC, see Overview of VPCs and subnets.
5. Choose one or more **VPC security groups**. For more information, see Control traffic to resources using security groups.
6. Choose whether to enable extra compute resources for automatic optimizations. For more information, see Allocating extra compute resources for automatic database optimization in the *Amazon Redshift Database Developer Guide*.
7. Under **Subnet**, specify one or more subnets to associate with your database. These subnets are contained in the Amazon VPC you chose previously and must be in three distinct Availability Zones. For more information, see Considerations when using Amazon Redshift Serverless.
8. Select the base RPU capacity that conforms with your requirements.

Choose a namespace

1. Choose either **Create a new namespace**, and enter the namespace name, or **Add to an existing namespace**, and select the namespace from the drop-down list.
2. For **Database name and password**, specify the name of the first database. You can also specify an admin other than your default console admin, by editing the **Admin user credentials**.
3. For **Permissions**, you choose **Associate IAM role** to associate specific IAM roles with your namespace and workgroup. For more information about associating IAM roles with Amazon Redshift, see Identity and access management in Amazon Redshift.
4. You can customize your encryption settings by creating a new key or choosing a key other than the default. For **Audit logging**, choose the logs to export. Each log type specifies different metadata. Choose **Continue** to review your choices.

Review workgroup selections

1. Review your settings under **Review and create**. It shows the settings you chose in the previous steps.
2. Choose **Save**.

After you create the workgroup, it's added to the **Workgroups** list.

Viewing properties for a workgroup

In Amazon Redshift Serverless, a workgroup is a collection of compute resources available for use. When you choose Amazon Redshift Serverless, in the AWS console, you can choose **Workgroup configuration** from the navigation menu to view a list. You can use the **Search** box to find workgroups that meet your search criteria. Each workgroup entry has a few properties displayed:

- **Workgroup** - The name of the workgroup. You can select it to view and edit the workgroup's properties.
- **Status** - Shows whether the workgroup is available.
- **Namespace** - The namespace associated with the workgroup. Each workgroup is associated with one namespace.
- **Creation date** - The date (UTC) that the workgroup was created.
- **Tags** - Tags associated with the workgroup.

Additionally, **Workgroup configuration** has another list for managed workgroups, which are Amazon Redshift Serverless workgroups managed by AWS Glue. For more information on managed workgroups, see Managed workgroups in the Amazon Redshift Database Developer Guide.

Workgroup properties

You can list workgroups by choosing **Workgroup configuration** in the left menu. Then you can choose a workgroup from the list. Several panels show properties for the workgroup. You can also perform actions. **General information** displays the following:

- **Workgroup** - The name of the workgroup.
- **Namespace** - The namespace associated with the workgroup. You can choose it to view its properties. A workgroup is associated with a single namespace.

- **Date created** - When the workgroup was created.
- **Status** - Indicates if the workgroup resources are available. If it's available, you can connect with a client to the Amazon Redshift Serverless instance, in order to query data or create database resources, or you can connect with query editor v2.
- **Endpoint** - The URL.
- **JDBC URL** - The URL to establish JDBC client connections. You can use this URL to connect with a JDBC driver for Amazon Redshift. For more information, see Configuring a connection for JDBC driver version 2.x for Amazon Redshift.
- **ODBC URL** - The URL to establish ODBC client connections. It contains properties, like the database and user ID, and their values.
- **Workgroup version and Patch version** - Amazon Redshift Serverless regularly releases new versions and patches. You can use the workgroup version and Patch version numbers to track software updates to your Amazon Redshift Serverless workgroup. For more information about changes and features in specific patches, see Cluster versions for Amazon Redshift.
- **Workgroup ARN** - The Amazon Resource Name for the workgroup.
- **Extra compute resources for automatic optimizations** - Whether Amazon Redshift is allocating extra compute resources to perform automatic optimizations, even during periods of heavy usage. For more information, see Allocating extra compute resources for automatic database optimization in the *Amazon Redshift Database Developer Guide*.

The **Data access** tab contains several panels:

- **Network and security** - You can see network properties, such as the **Virtual private cloud (VPC)** identifier, **VPC security group** list, **Enhanced VPC routing**, **IP address type**, and the **Publicly accessible** setting. If you choose **Edit**, you can change these settings. Additionally, you can select **Turn on enhanced VPC routing**, which routes network traffic between your serverless database and data repositories through a VPC, for enhanced privacy and security. You can also select **Turn on Public Accessible**, which makes the database publicly accessible from outside the VPC, allowing instances and devices to connect.

  The **IP address type** can be set to dual-stack mode to support access to workgroups on both IPv4 an IPv6 at the same time. For more information about network layer communications Internet Protocols (IP), see Internet Protocol in *Wikipedia*.
- **Redshift managed VPC endpoints** - You can create managed VPC endpoints to access Amazon Redshift Serverless from another VPC.

The **Limits** tab has settings for controlling capacity and use limits for Amazon Redshift Serverless. It contains the following panels:

- **Base capacity in Redshift processing units (RPUs)** - You can set the base capacity of the compute resources used to process your workload. For more information, see Compute capacity for Amazon Redshift Serverless.
- **Usage limits** - You can set up to four limits for the maximum compute resources that your Amazon Redshift Serverless instance can use in a time period, and select actions for Amazon Redshift Serverless to perform when reaching those limits. For example, you can set your workgroup to have two limits, one of 500 RPU hours and one of 900 RPU hours. You can have Amazon Redshift Serverless send you an alert when it reaches the first limit of 500 RPU hours, then turn off user queries when it reaches the second limit of 900 hours. These limits help control your costs and make them more predictable.
- **Query limits** - You can set limits on queries, like the timeout setting. These limits help you optimize cost and performance.

The **Tabs** tab has the **Tags** panel, which shows any tags that you created for your workgroup. For more information about tagging resources, see Tagging resources in Amazon Redshift Serverless.

Managed workgroup properties

You can also choose workgroups managed by the AWS Glue Data Catalog under the **Managed workgroups** list.

Managed workgroups have different properties from regular workgroups. For more information on managed workgroups, see Managed workgroups in the Amazon Redshift Database Developer Guide.

General information displays the following:

- **Workgroup** - The name of the managed workgroup.
- **Date created** - The date (UTC) that the managed workgroup was created.
- **Catalog ARN** - The Amazon Resource Name (ARN) for the managed workgroup in the AWS Glue Data Catalog.
- **Status** - Indicates if the managed workgroup's compute resources are available. If the resources are available, you can connect to the catalog that uses the managed workgroup with an Apache Iceberg-compatible SQL client in order to query data or create database resources. You can also connect to the catalog using the Amazon Redshift query editor v2.

Query and database monitoring contains the **Managed workgroup performance** graph, showing the average elapsed time of all queries from the workgroup over time.

The **Query history** tab is a list of all queries from the managed workgroup. Its details include information such as the user who ran the query, the client engine from which the query originated, and the query's ID and status. The Users tab is a list of all the users in the workgroup. The **Performance metrics** tab shows various metrics such as average query time, number of completed queries, and percentage of storage capacity used.

Deleting a workgroup

You can delete a workgroup using the console. Before you do this, make sure that you have your data backed up and snapshots in place. Resources deleted as part of the workgroup in many cases can't be retrieved.

Complete the following steps:

1. Choose **Amazon Redshift Serverless**, choose **Workgroup configuration** and choose **Delete Amazon Redshift Serverless instance**.

2. A dialogue opens. When you choose to delete the workgroup, all usage limits are removed, all VPC endpoints are removed, and access to VPC endpoints is removed.

   Type *delete* and select **Delete** to confirm.

After you complete the steps, the status of the workgroup is *Deleting* and a banner indicates that the workgroup is being deleted. While the delete process is in progress, some features under the **Serverless dashboard** are disabled. But you can configure provisioned clusters on the **Provisioned clusters dashboard**.

After you delete the workgroup, it doesn't appear with the namespace. You can choose the **Create workgroup** button to create a new one.

You can delete an existing workgroup and associate a new workgroup with a different configuration to the same namespace. When creating the new workgroup, choose the base capacity that works with the size of the data associated with the namespace.

You can associate a workgroup with a namespace that was created with a customer-managed key (CMK). For more information about AWS KMS, see AWS KMS concepts.

Namespaces

In Amazon Redshift Serverless, a namespace defines a logical container for database objects. It can hold tables, workgroups, and other database resources. If you haven't created a workgroup and a namespace, and you are looking for instructions in how to get started with Amazon Redshift Serverless, see Setting up Amazon Redshift Serverless for the first time.

Namespace properties

In Amazon Redshift Serverless, a namespace defines a container for database objects. You can choose **Namespace configuration** from the navigation list, choose a namespace from the list, and edit its settings.

General information for the namespace includes the following:

- **Namespace** - The name.
- **Namespace ID** - The unique identifier.
- **ARN** - A unique identifier used to specify the resource across AWS. It contains properties like the region and the service.
- **Status** - The status, such as **Available**.
- **Date created** - The date (UTC) that the namespace was created.
- **Storage used** - The storage space used by the namespace and all of its objects.
- **Admin user name** - The admin account. This is typically the account used to create the namespace.
- **Database name** - The name of the database contained by the namespace.
- **Total table count** - The count of tables in all schemas.

Additional settings and properties for the namespace are on several tabs. These include the following:

- **Workgroup** - Shows the workgroup associated with the namespace.
- **Data back up** - On this panel, you can configure and create snapshots, and configure recovery points.
- **Security and encryption** - You can manage IAM role permissions and view or edit your security and encryption settings. These include your encryption key status, and the setting to turn on

audit logging. For more information about audit logging for Amazon Redshift Serverless, see Audit logging for Amazon Redshift Serverless.

- **Datashares** - Shows datashares. With data sharing, you can provide access to data without the need to copy it or move it. For more information about data sharing, see Data sharing in Amazon Redshift Serverless.

Searching for a namespace

From the Amazon Redshift menu, you can choose from the **Namespaces** list in order to view or edit the properties for a namespace. Information on the console includes the namespace name, the admin name, and other properties.

A namespace's settings and properties are on several tabs. These include the following:

- **Workgroup** - Shows workgroups associated with the namespace.
- **Data back up** - You can configure and create snapshots, and configure recovery points.
- **Security and encryption** - You can manage IAM role permissions and view or edit your security and encryption settings. These include your encryption key status and your audit logging settings.
- **Datashares** - Shows datashares.

Editing security and encryption

Amazon Redshift Serverless is secured by means of KMS encryption. You can update encryption settings via the console:

1. Choose **Namespace configuration** from the main menu on the console, choose the namespace to edit, and choose **Edit** on the **Security and encryption** tab. A dialog appears.
2. You can select **Customize encryption settings** and then **Choose an AWS customer managed key** to change the key used to encrypt your resources.
3. For **Audit logging**, choose the logs to export. Each log type specifies different metadata.
4. To complete the configuration update, choose **Save changes**.

Changing the AWS KMS key for a namespace

In Amazon Redshift, encryption protects data at rest. Amazon Redshift Serverless uses AWS KMS key encryption automatically to encrypt both your Amazon Redshift Serverless resources and snapshots. As a best practice, most organizations review the type of data they store and have a plan to rotate encryption keys on a schedule. The frequency for rotating keys can vary, depending on your policies for data security. Amazon Redshift Serverless supports changing the AWS KMS key for the namespace so you can adhere to your organization's security policies.

When you change the AWS KMS key, the data remains unchanged.

Changing an AWS KMS key using the console

In Amazon Redshift, encryption protects data at rest. Amazon Redshift Serverless uses AWS KMS key encryption automatically to encrypt both Amazon Redshift Serverless and snapshots. As a best practice, most organizations review the type of data they store and have a plan to rotate encryption keys on a schedule. The frequency for rotating keys can vary, depending on your policies for data security. Amazon Redshift Serverless supports changing the AWS KMS key for the namespace so you can adhere to your organization's security policies.

When you change the AWS KMS key, the data remains unchanged.

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Namespace configuration**. Choose your namespace from the list.
3. From the **Security and encryption** tab, choose **Edit**.
4. Choose **Customize encryption settings** and then choose a key for the namespace. You can optionally create a new key.

Changing AWS KMS encryption keys using the AWS CLI

Use `update-namespace` to change the AWS KMS key for the namespace. The following shows the syntax for the command:

```
aws redshift-serverless update-namespace
--namespace-name
[--kms-key-id <id-of-kms-key>]
```

```
// other parameters omitted here
```

You must have a namespace created or the CLI command results in an error.

The time it takes to change the key depends on the amount of data in Amazon Redshift Serverless. This typically takes fifteen minutes per 8TB of stored data.

Limitations

You can't change from a customer managed KMS Key to an AWS KMS key. In this case, you have to create a new namespace.

You can't perform other actions while the key is being changed.

Deleting a namespace

If you want to delete a namespace with an associated workgroup, you have to first delete the workgroup.

On the Amazon Redshift Serverless console, complete the following steps:

1. Choose **Namespace configuration** from the left menu and then choose the namespace you want to delete from the list.
2. Choose **Actions** and select **Delete namespace**.
3. A dialogue box opens. You can keep your data by creating a manual snapshot prior to completing the delete operation.

   Type *delete* and select **Delete** to confirm.

Monitoring queries and workloads with Amazon Redshift Serverless

You can monitor your Amazon Redshift Serverless queries and workload with the provided system views.

Monitoring views are system views in Amazon Redshift Serverless that are used to monitor query and workload usage. These views are located in the `pg_catalog` schema. The system views available have been designed to give you the information needed to monitor Amazon Redshift Serverless, which is much simpler than that needed for provisioned clusters. The SYS system views

have been designed to work with Amazon Redshift Serverless. To display the information provided by these views, run SQL SELECT statements.

System views are defined to support the following monitoring objectives.

Workload monitoring

You can monitor your query activities over time to:

- Understand workload patterns, so you know what is normal (baseline) and what is within business service level agreements (SLAs).
- Rapidly identify deviation from normal, which might be a transient issue or something that warrants further action.

Data load and unload monitoring

Data movement in and out of Amazon Redshift Serverless is a critical function. You use COPY and UNLOAD to load or unload data, and you must monitor progress closely in terms of bytes/rows transferred and files completed to track adherence to business SLAs. This is normally done by running system table queries frequently (that is, every minute) to track progress and raise alerts for investigation/corrective action if significant deviations are detected.

Failure and problem diagnostics

There are cases where you must take action for query or runtime failures. Developers rely on system tables to self-diagnose issues and determine correct remedies.

Performance tuning

You might need to tune queries that are not meeting SLA requirements either from the start, or have degraded over time. To tune, you must have runtime details including run plan, statistics, duration, and resource consumption. You need baseline data for offending queries to determine the cause for deviation and to guide you how to improve performance.

User objects event monitoring

You need to monitor actions and activities on user objects, such as refreshing materialized views, vacuum, and analyze. This includes system-managed events like auto-refresh for materialized views. You want to monitor when an event ends if it is user initiated, or the last successful run if system initiated.

Usage tracking for billing

You can monitor your usage trends over time to:

- Inform budget planning and business expansion estimates.
- Identify potential cost-saving opportunities like removing cold data.

Use the SYS system views to monitor Amazon Redshift Serverless;. For more information about the SYS monitoring views, go to SYS monitoring views in the Amazon Redshift Database Developer Guide.

Adding a query monitoring policy

A superuser can provide access to users who aren't superusers so that they can perform query monitoring for all users. First, you add a policy for a user or a role to provide query monitoring access. Then, you grant query monitoring permission to the user or role.

To add the query monitoring policy

1. Choose https://console.aws.amazon.com/iam/.
2. Under **Access management**, choose **Policies**.
3. Choose **Create Policy**.
4. Choose **JSON** and paste the following policy definition.

   JSON

```
{
"Version":"2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
    "redshift-data:ExecuteStatement",
    "redshift-data:DescribeStatement",
    "redshift-data:GetStatementResult",
    "redshift-data:ListDatabases"
],
"Resource": "*"
},
{
"Effect": "Allow",
"Action": "redshift-serverless:GetCredentials",
"Resource": "*"
```

```
}
]
}
```

5. Choose **Review policy**.
6. For **Name**, enter a name for the policy, such as `query-monitoring`.
7. Choose **Create policy**.

After you create the policy, you can grant the appropriate permissions.

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.
- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Create a role for a third-party identity provider (federation) in the *IAM User Guide*.
- IAM users:
  - Create a role that your user can assume. Follow the instructions in Create a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

Granting query monitoring permissions for a user

Users with `sys:monitor` permission can view all queries. In addition, users with `sys:operator` permission can cancel queries, analyze query history, and perform vacuum operations.

To grant query monitoring permission for a user

1. Enter the following command to provide system monitor access, where *user-name* is the name of the user for whom you want to provide access.

   ```
   grant role sys:monitor to "IAM:user-name";
   ```

2. (Optional) Enter the following command to provide system operator access, where *user-name* is the name of the user for whom you want to provide access.

```
grant role sys:operator to "IAM:user-name";
```

Granting query monitoring permissions for a role

Users with a role that has `sys:monitor` permission can view all queries. In addition, users with a role that has `sys:operator` permission can cancel queries, analyze query history, and perform vacuum operations.

To grant query monitoring permission for a role

1. Enter the following command to provide system monitor access, where *role-name* is the name of the role for which you want to provide access.

```
grant role sys:monitor to "IAMR:role-name";
```

2. (Optional) Enter the following command to provide system operator access, where *role-name* is the name of the role for which you want to provide access.

```
grant role sys:operator to "IAMR:role-name";
```

Setting usage limits, including setting RPU limits

Under the **Limits** tab for a workgroup, you can add one or more usage limits to control the maximum RPUs you use in a given time period, or to set a data sharing usage limit.

1. Choose **Manage usage limits**. The limits section appears at the bottom of the **Compute usage by period** panel.
2. Set a usage limit in number of RPU hours.
3. Choose a **Frequency**, which is either **Daily**, **Weekly**, or **Monthly**. This sets the time period for the usage limit. Choosing **Daily** in this instance gives you more detailed control.
4. Set a usage limit, in number of hours.
5. Set the action. These are the following:

- **Log to system table** - Adds a record to the system view SYS_QUERY_HISTORY. You can query the `usage_limit` column in this view to determine if a query exceeded the limit.
- **Alert** - Uses Amazon SNS to set up notification subscriptions and send notifications if a limit is breached. You can choose an existing Amazon SNS topic or create a new one.
- **Turn off user queries** - Disables queries to stop use of Amazon Redshift Serverless. It also sends a notification.

  The first two actions are informational, but the last turns off query processing.

6. Optionally, you can set a **Cross-Region data sharing usage limit**, which limits how much data transferred from producer Region to consumer Region consumers can query. To do this, choose **Add limit** and follow the steps.
7. Choose **Save changes** at the bottom of the page to save the limit.
8. Set up to 3 more limits as necessary.

For more conceptual information about RPUs and billing, see Billing for Amazon Redshift Serverless.

Setting query limits

Under the **Limits** tab for a workgroup, you can add a limit to monitor performance and limits. For more information about query monitoring limits, see WLM query monitoring rules.

1. Choose **Manage query limits**. Choose **Add new limit** on the **Manage query limits** dialogue.
2. Choose the limit type you want to set and enter a value for its corresponding limit.
3. Choose **Save changes** to save the limit.

When you change your query limit and configuration parameters, your database will restart.

Setting query queues

Amazon Redshift Serverless supports queue-based query resource management. You can create dedicated query queues with customized monitoring rules for different workloads. This feature provides granular control over resource usage.

Query monitoring rules (QMR) apply only at the Redshift Serverless workgroup level, affecting all queries run in this workgroup uniformly. Queue-based approach lets you create queues with distinct monitoring rules. You can assign these queues to specific user roles and query groups. Each queue operates independently, with rules affecting only the queries within that queue.

Queues let you set metrics-based predicates and automated responses. For example, you can configure rules to automatically abort queries that exceed time limits or consume too many resources.

Considerations

Consider the following when using serverless queues:

- The following Workload Management (WLM) configuration keys used in Amazon Redshift provisioned clusters are not supported in Redshift Serverless queues: `max_execution_time`, `short_query_queue`, `auto_wlm`, `concurrency_scaling`, `priority`, `queue_type`, `query_concurrency`, `memory_percent_to_use`, `user_group`, `user_group_wild_card`.

  Additionally hop, change_query_priority actions are not supported in Serverless.
- The hop Action (moving queries between queues) is not supported in Amazon Redshift Serverless.
- Queue priorities are supported only for Amazon Redshift provisioned clusters.
- Amazon Redshift Serverless automatically manages scaling and resource allocation for optimal performance, so you don't need to manually configure queue priorities.

Setting up query queues

You can create queues under the Limits tab for a serverless workgroup using the AWS Management Console, AWS CLI, or Redshift Serverless API.

Console

Follow these steps to create a queue for your serverless workgroup.

1. Navigate to your Redshift Serverless workgroup.
2. Select the Limits tab.
3. Under **Query Queues**, choose **Enable Queues**.

> **Important**
>
> Enabling query queues is a permanent change. You cannot revert to queue-less monitoring once enabled.

4. Configure your queues using the following parameters:

Queue level parameters

- `name` - Queue identifier (required, unique, non-empty)
- `user_role` - Array of user roles (optional)
- `query_group` - Array of query groups (optional)
- `query_group_wild_card` - 0 or 1 to enable wildcard matching (optional)
- `user_group_wild_card` - 0 or 1 to enable wildcard matching (optional)
- `rules` - Array of monitoring rules (optional)

Rule level parameters

- `rule_name` - Unique identifier, max 32 chars (required)
- `predicate` - Array of conditions, 1-3 predicates (required)
- `action` - "abort" or "log" (required)

Predicate level parameters

- `metric_name` - Metric to monitor (required)
- `operator` - "=", "<", or ">" (required)
- `value` - Numeric threshold (required)

Limits

- Max 8 queues
- Max 25 rules across all queues
- Max 3 predicates per rule
- Rule names must be unique globally

Example Configuration

Three queues example: one for dashboarding queries with a short timeout, one for ETL queries with a long timeout and an admin queue:

```
[
  {
    "name": "dashboard",
    "user_role": ["analyst", "viewer"],
    "query_group": ["reporting"],
    "query_group_wild_card": 1,
    "rules": [
      {
        "rule_name": "short_timeout",
        "predicate": [
          {
            "metric_name": "query_execution_time",
            "operator": ">",
            "value": 60
          }
        ],
        "action": "abort"
      }
    ]
  },
  {
    "name": "ETL",
    "user_role": ["data_scientist"],
    "query_group": ["analytics", "ml"],
    "rules": [
      {
        "rule_name": "long_timeout",
        "predicate": [
          {
            "metric_name": "query_execution_time",
            "operator": ">",
            "value": 3600
          }
        ],
        "action": "log"
      },
      {
        "rule_name": "memory_limit",
        "predicate": [
```

```
          {
            "metric_name": "query_temp_blocks_to_disk",
            "operator": ">",
            "value": 100000
          }
        ],
        "action": "abort"
      }
    ]
  },
  {
    "name": "admin_queue",
    "user_role": ["admin"],
    "query_group": ["admin"]
  }
]
```

In this example:

- Dashboard queries are aborted if they run more than 60 seconds
- ETL queries are logged if they run more than an hour
- Admin queue does not have any resource limits

CLI

You can manage queues using the CreateWorkgroup or UpdateWorkgroup APIs with the `wlm_json_configuration` config parameter to specify queues in JSON format.

```
aws redshift-serverless create-workgroup \
  --workgroup-name test-workgroup \
  --namespace-name test-namespace \
  --config-parameters '[{"parameterKey": "wlm_json_configuration", "parameterValue":
 "[{\"name\":\"dashboard\",\"user_role\":[\"analyst\",\"viewer\"],\"query_group
\":[\"reporting\"],\"query_group_wild_card\":1,\"rules\":[{\"rule_name\":
\"short_timeout\",\"predicate\":[{\"metric_name\":\"query_execution_time\",
\"operator\":\">\",\"value\":60}],\"action\":\"abort\"}]},{\"name\":\"ETL
\",\"user_role\":[\"data_scientist\"],\"query_group\":[\"analytics\",\"ml
\"],\"rules\":[{\"rule_name\":\"long_timeout\",\"predicate\":[{\"metric_name
\":\"query_execution_time\",\"operator\":\">\",\"value\":3600}],\"action\":
\"log\"},{\"rule_name\":\"memory_limit\",\"predicate\":[{\"metric_name\":
\"query_temp_blocks_to_disk\",\"operator\":\">\",\"value\":100000}],\"action\":
```

```
\"abort\"}]},{\"name\":\"admin_queue\",\"user_role\":[\"admin\"],\"query_group\":
[\"admin\"]}]"}]'
```

Best practices

Keep the following best practices in mind when you use serverless queues.

- Use separate queues for workloads with distinct limit requirements (e.g., ETL, reporting, or ad-hoc analysis).
- Start with simple thresholds and adjust based on query behavior and usage patterns. You can monitor query usage patterns using the tables and views documented in System tables and views for query monitoring rules.

Checking Amazon Redshift Serverless summary data using the dashboard

The Amazon Redshift Serverless dashboard contains a collection of panels that show at-a-glance metrics and information about your workgroup and namespace. These panels include the following:

- **Resources summary** - Displays high-level information about Amazon Redshift Serverless, such as the storage used and other metrics.
- **Query summary** - Displays information about queries, including completed queries and running queries. Choose **View details** to go to a screen that has additional filters.
- **RPU capacity used** - Displays the overall capacity used over a given time period, like the previous ten hours, for instance.
- **Datashares** - Shows the count of datashares, which are used to share darta between, for example, AWS accounts. The metrics show which datashares require authorization, and other information.
- **Total compute usage** - Shows your total consumed RPU hours for the selected workgroup over a selected time range, up to the last 7 days.

From the dashboard you can quickly dive into these available metrics to check a detail regarding Amazon Redshift Serverless, or review queries, or track work items.

Audit logging for Amazon Redshift Serverless

You can configure Amazon Redshift Serverless to export connection, user, and user-activity log data to a log group in Amazon CloudWatch Logs. With Amazon CloudWatch Logs, you can perform real-time analysis of the log data and use CloudWatch to create alarms and view metrics. You can use CloudWatch Logs to store your log records in durable storage.

You can create CloudWatch alarms to track your metrics using the Amazon Redshift console. For more information on creating alarms, see Managing alarms.

To export generated log data to Amazon CloudWatch Logs, the respective logs must be selected for export in your Amazon Redshift Serverless configuration settings, on the console. You can do this by choosing the **Namespace configuration** settings, under **Security and encryption**.

Log events in CloudWatch

After selecting which Redshift logs to export, you can monitor events in Amazon CloudWatch Logs. A new log group is automatically created for Amazon Redshift Serverless, in which `log_type` represents the log type.

```
/aws/redshift/<namespace>/<log_type>
```

When you create your first workgroup and namespace, *default* is the namespace name. The log group name varies according to what you call the namespace.

For example, if you export the connection log, log data is stored in the following log group.

```
/aws/redshift/default/connectionlog
```

Log events are exported to a log group using the serverless log stream. The behavior depends on which of the following conditions are true:

- **A log group with the specified name exists.** Redshift exports log data using the existing log group. To create log groups with predefined log-retention periods, metric filters, and customer access, you can use automated configuration, such as that provided by **AWS CloudFormation**.
- **A log group with the specified name doesn't exist.** When a matching log entry is detected in the log for the instance, Amazon Redshift Serverless creates a new log group in Amazon CloudWatch Logs automatically. The log group uses the default log-retention period of *Never Expire*. To change the log-retention period, use the Amazon CloudWatch Logs console, the AWS

CLI, or the Amazon CloudWatch Logs API. For more information about changing log-retention periods in CloudWatch Logs, see *Change log data retention* in Working with log groups and log streams.

To search for information within log events, use the Amazon CloudWatch Logs console, the AWS CLI, or the Amazon CloudWatch Logs API. For more information about searching and filtering log data, see Searching and filtering log data.

CloudWatch metrics

Amazon Redshift Serverless metrics are divided into compute metrics and data and storage metrics, falling under the workgroup and namespace dimension sets, respectively. For more information about workgroups and namespaces, see Workgroups and namespaces.

CloudWatch compute metrics are the following:

Metric name	Units	Description	Dimension sets
`QueriesCompletedPerSecond`	Number of queries	The number of queries completed each second.	{Database, LatencyRange, Workgroup}, {LatencyRange, Workgroup}
`QueryDuration`	Microseconds	The average amount of time to complete a query.	{Database, LatencyRange, Workgroup}, {LatencyRange, Workgroup}
`QueriesRunning`	Number of queries	The number of running queries at a point in time.	{Database , QueryType , Workgroup }, {QueryType, Workgroup}
`QueriesQueued`	Number of queries	The number of queries in	{Database , QueryType

Metric name	Units	Description	Dimension sets
		the queue at a point in time.	, Workgroup }, {QueryType, Workgroup}
DatabaseConnections	Number of connections	The number of connections to a database at a point in time.	{Database, Workgroup}, {Workgroup}
QueryRuntimeBreakdown	Milliseconds	The total time queries ran, by query stage.	{Database, Stage, Workgroup}, {Stage, Workgroup}
ComputeCapacity	RPU	Average number of compute units allocated during the past 30 minutes, rounded up to the nearest integer.	{Workgroup}
ComputeSeconds	RPU-seconds	Accumulated compute-unit seconds used in the last 30 minutes.	{Workgroup}
QueriesSucceeded	Number of queries	The number of queries that succeeded in the last 5 minutes.	{Database , QueryType , Workgroup }, {QueryType, Workgroup}

Metric name	Units	Description	Dimension sets
QueriesFailed	Number of queries	The number of queries that failed in the last 5 minutes.	{Database , QueryType , Workgroup }, {QueryType, Workgroup}

Metric name	Units	Description	Dimension sets
UsageLimitAvailable	RPU-hours or TBs	Depending on the UsageType, UsageLimitAvailable returns the following: • If the UsageType is SERVERLESS_COMPUTE, UsageLimitAvailable returns the remaining number of RPU-hours that the workgroup can query in the given limit. • If the UsageType is CROSS_REGION_DATASHARING, UsageLimitAvailable returns the remaining number of TBs that the customer can	{UsageLimitId, UsageType, Workgroup}

Metric name	Units	Description	Dimension sets
		scan in the given limit.	

Metric name	Units	Description	Dimension sets
UsageLimitConsumed	RPU-hours or TBs	Depending on the UsageType, UsageLimitConsumed returns the following: • If the UsageType is SERVERLESS_COMPUTE, UsageLimitConsumed returns the number of RPU-hours that the workgroup has already queried in the given limit. • If the UsageType is CROSS_REGION_DATASHARING, UsageLimitConsumed returns the number of TBs that the customer has already used	{UsageLimitId, UsageType, Workgroup}

Metric name	Units	Description	Dimension sets
		to scan in the given limit.	
ExtraComputeForAutomaticOptimizationChargedSeconds	RPU-seconds	Number of compute-unit seconds charged for automatic optimization operations in the last 30 minutes.	{Workgroup}

CloudWatch data and storage metrics are the following:

Metric name	Units	Description	Dimension sets
TotalTableCount	Number of tables	The total number of tables at a particular point in time. This total includes permanent tables, temporary tables, datashare tables, external tables, and materialized views. Views and system tables are not	{Database, Namespace}

Metric name	Units	Description	Dimension sets
		included in this count.	
DataStorage	Megabytes	The number of megabytes used, in disk or storage space, for Redshift data.	{Namespace}

The ServerlessUniqueSnapshotStorage metric is namespace- and workgroup-agnostic. CloudWatch's ServerlessUniqueSnapshotStorage metric is as follows:

Metric name	Units	Description	Dimension sets
ServerlessUniqueSnapshotStorage	Megabytes	The number of megabytes of unique data blocks stored across all active manual snapshots.	{}

Dimension sets are the grouping dimensions applied to your metrics. You can use these dimension groups to specify how your statistics are retrieved.

The following table details dimensions and dimension values for specific metrics:

Dimension	Description and values
DatabaseName	The name of the database. A custom value.
Latency	Possible values are as follows: • Short – under 10 seconds

Dimension	Description and values
	• Medium – between 10 seconds and 10 minutes • Long – over 10 minutes
QueryType	Possible values are INSERT, DELETE, UPDATE, UNLOAD, LOAD, SELECT, CTAS, and OTHER.
stage	The execution stages for a query. Possible values are as follows: • QueryPlanning: Time spent parsing and optimizing SQL statements. • QueryWaiting: Time spent waiting in the WLM queue. • QueryExecutingRead: Time spent executing read queries. • QueryExecutingInsert: Time spent executing insert queries. • QueryExecutingDelete: Time spent executing delete queries. • QueryExecutingUpdate: Time spent executing update queries. • QueryExecutingCtas: Time spent executing create table as queries. • QueryExecutingUnload: Time spent executing unload queries. • QueryExecutingCopy: Time spent executing copy queries. • QueryCommit: Time spent committing.
Namespace	The name of the namespace. A custom value.
Workgroup	The name of the workgroup. A custom value.

Dimension	Description and values
`UsageLimitId`	The identifier of the usage limit.
`UsageType`	The Amazon Redshift Serverless feature being limited. Possible values are as follows: • SERVERLESS_COMPUTE • CROSS_REGION_DATASHARING

Snapshots and recovery points

A backup in Amazon Redshift Serverless is a point-in-time representation of the objects and data in your namespace. There are two types of backups: snapshots that are manually created and recovery points that Amazon Redshift Serverless automatically creates for you.

Amazon Redshift Serverless automatically creates recovery points every 30 minutes or after every 5 GB of data changes per node, whichever happens first. For larger datasets (more than 5 GB × number of nodes), the minimal interval between recovery points is 15 minutes. All recovery points are kept for 24 hours.

> **Note**
>
> You cannot create your own snapshot schedule to control when recovery points are created.

Amazon Redshift Serverless creates snapshots in Redshift Managed Storage (RMS). For more information, see Compute capacity for Amazon Redshift Serverless.

> **Note**
>
> No-backup tables aren't supported for RG or RA3 provisioned clusters and Amazon Redshift Serverless workgroups. A table marked as no-backup in an RG or RA3 cluster or serverless workgroup is treated as a permanent table that will always be backed up while taking a snapshot, and always restored when restoring from a snapshot. To avoid snapshot costs for no-backup tables, truncate them before taking a snapshot.

If you find that you want to retrieve the data in a snapshot or a recovery point, you can restore a snapshot to a serverless namespace or to a provisioned cluster. There are three scenarios in which you can restore snapshots:

- Restore a serverless snapshot to a serverless namespace.
- Restore a serverless snapshot to a provisioned cluster.
- Restore a provisioned cluster snapshot to a serverless namespace.

When you restore a serverless snapshot to a provisioned cluster, you must choose the node type to use, such as RG or RA3, and the number of nodes, letting you control settings at the cluster or node level.

To restore a provisioned cluster snapshot to a serverless namespace, start from the Redshift provisioned console, choose the snapshot to restore, then choose **Restore from snapshot**, **Restore to serverless namespace**. Amazon Redshift converts tables with interleaved keys into compound sort keys when you restore a provisioned cluster snapshot to a serverless namespace. For more information about sort keys, see Working with sort keys.

If you want to add additional context, you can tag snapshots and recovery points with key-value pairs that provide metadata and information to snapshots and recovery points. For more information about tagging resources, see Tagging resources overview.

Finally, you can also share snapshots with other AWS accounts, which lets them access data within the snapshot and run queries.

AWS Backup integration

You can also create and restore snapshots using AWS Backup, a fully managed service that helps you centralize and automate data protection across AWS services, in the cloud, and on premises. For more information, see AWS Backup integration with Amazon Redshift. For information on AWS Backup, see What is AWS Backup? in the *AWS Backup Developer Guide*.

Creating a snapshot

> **Note**
>
> No-backup tables aren't supported for RG or RA3 provisioned clusters and Amazon Redshift Serverless workgroups. A table marked as no-backup in an RG or RA3 cluster or serverless

workgroup is treated as a permanent table that will always be backed up while taking a snapshot, and always restored when restoring from a snapshot. To avoid snapshot costs for no-backup tables, truncate them before taking a snapshot.

To create a snapshot, perform the steps in the following procedure.

To create a snapshot

1. On the Amazon Redshift Serverless console, choose **Data backup**.
2. Choose **Create snapshot**.
3. Choose a namespace to create a snapshot of.
4. Enter a snapshot identifier.
5. (Optional) Choose a retention period. If you choose **Custom value**, choose the number of days. The amount you choose must be between 1-3653 days, inclusive. The default is retain indefinitely.
6. Choose **Create**.

To create a snapshot from namespace configuration

1. On the Amazon Redshift Serverless console, choose **Namespace configuration**.
2. Choose the namespace to create a snapshot of. You can only create snapshots of namespaces that are associated with a workgroup and whose statuses are Available.
3. Choose the **Data backup** tab.
4. Choose **Create snapshot**.
5. Enter a snapshot identifier.
6. (Optional) Choose a retention period. If you choose **Custom value**, choose the number of days. The amount you choose must be between 1-3653 days, inclusive.
7. Choose **Create**.

Creating a final snapshot

To create a final snapshot of all data within a namespace before deleting the namespace, perform the steps in the following procedure.

To create a final snapshot

1. On the Amazon Redshift Serverless console, choose **Namespace configuration**.
2. Choose the namespace to delete.
3. Choose **Actions**, **Delete**.
4. Choose **Create final snapshot**.
5. Enter a name for the snapshot.
6. Enter delete.
7. Choose **Delete**.

Sharing a snapshot or removing snapshot permissions

To share a snapshot with another AWS account or remove an account's access to a snapshot, perform the following procedure.

To share or remove access to a snapshot

1. On the Amazon Redshift Serverless console, choose **Data backup**.
2. Choose a snapshot to share.
3. Choose **Actions**, **Manage access**.
4. To share a snapshot with another account, enter an **AWS account ID**. To remove access from an account, choose **Remove**.
5. Choose **Save changes**.

Scheduling a snapshot

To precisely control when to take a snapshot, you can create a snapshot schedule for specific namepsaces. When scheduling snapshot creation, you can create either a one-time event or use Unix cron expressions to create a recurring schedule. Cron expressions support three fields and are separated by white space.

```
cron(Minutes Hours Day-of-month Month Day-of-week Year)
```

Fields	Values	Wildcards
Minutes	0–59	, - * /
Hours	0–23	, - * /
Day-of-month	1–31	, - * ? / L W
Month	1–12 or JAN-DEC	, - * /
Day-of-week	1–7 or SUN-SAT	, - * ? L #
Year	1970–2199	, - * /

Wildcards

- The **,** (comma) wildcard includes additional values. In the `Day-of-week` field, `MON,WED,FRI` would include Monday, Wednesday, and Friday. Total values are limited to 24 per field.
- The **-** (dash) wildcard specifies ranges. In the `Hour` field, 1–15 would include hours 1 through 15 of the specified day.
- The ***** (asterisk) wildcard includes all values in the field. In the `Hours` field, ***** would include every hour.
- The **/** (forward slash) wildcard specifies increments. In the `Hours` field, you could enter **1/10** to specify every 10th hour, starting from the first hour of the day (for example, the 01:00, 11:00, and 21:00).
- The **?** (question mark) wildcard specifies one or another. In the `Day-of-month` field you could enter **7**, and if you didn't care what day of the week the seventh was, you could enter **?** in the Day-of-week field.
- The **L** wildcard in the `Day-of-month` or `Day-of-week` fields specifies the last day of the month or week.
- The **W** wildcard in the `Day-of-month` field specifies a weekday. In the `Day-of-month` field, `3W` specifies the day closest to the third weekday of the month.
- The **#** wildcard in the Day-of-week field specifies a certain instance of the specified day of the week within a month. For example, 3#2 would be the second Tuesday of the month: the 3 refers to Tuesday because it is the third day of each week, and the 2 refers to the second day of that type within the month.

> **Note**
>
> If you use a '#' character, you can define only one expression in the day-of-week field. For example, "3#1,6#3" is not valid because it is interpreted as two expressions.

Limits

- You can't specify the `Day-of-month` and `Day-of-week` fields in the same cron expression. If you specify a value in one of the fields, you must use a **?** (question mark) in the other.
- Snapshot schedules don't support the following frequencies:
  - Snapshots scheduled more frequently than 1 per hour.
  - Snapshots scheduled less frequently than 1 per day (24 hours).

  If you have overlapping schedules that result in scheduling snapshots within a 1 hour window, a validation error results.

The following table has some sample cron strings.

Minutes	Hours	Day of week	Meaning
0	14-20/1	TUE	Every hour between 2pm and 8pm on Tuesday.
0	21	MON-FRI	Every night at 9pm Monday–Friday.
30	0/6	SAT-SUN	Every 6 hour increment on Saturday and Sunday starting at 30 minutes after midnight (00:30) that day. This results in a

Minutes	Hours	Day of week	Meaning
			snapshot at [00:30, 06:30, 12:30, and 18:30] each day.
30	12/4	*	Every 4 hour increment starting at 12:30 each day. This resolves to [12:30, 16:30, 20:30].

The following example demonstrates how to create a schedule that runs in 2-hour increments starting at 15:15 each day.

```
cron(15 15/2 *)
```

You can use the Amazon Redshift Serverless console, API, or AWS CLI to create a snapshot schedule.

To schedule a snapshot

1. On the Amazon Redshift Serverless console, choose **Data backup**.
2. Choose **Snapshot schedules**.
3. Choose **Create schedule**.
4. Enter a name for the snapshot schedule.
5. Select the namespace to create snapshots for.
6. Enter a cron expression for the schedule or use the schedule builder to create one.
7. (Optional) Choose a retention period. If you choose **Custom value**, specify the number of days.
8. Choose **Create schedule**.

Updating a snapshot retention period

To update a snapshot retention period, perform the following procedure.

To update a snapshot retention period

1. On the Amazon Redshift Serverless console, choose **Data backup**.
2. Choose a snapshot to update.
3. Choose **Actions**, **Set manual snapshot settings.**
4. Choose a retention period. If you choose **Custom value**, choose the number of days.
5. Choose **Save changes**.

Deleting a snapshot

To delete a snapshot, perform the following procedure.

To delete a snapshot

> **Note**
>
> You can't delete a snapshot that's been shared with another account. You must first remove that account's access to the snapshot before deleting the snapshot.

1. On the Amazon Redshift Serverless console, choose **Data backup**.
2. Choose a snapshot to delete.
3. Choose **Actions**, **Delete**.
4. Choose **Delete**.

Restoring a snapshot

> **Note**
>
> No-backup tables aren't supported for RG or RA3 provisioned clusters and Amazon Redshift Serverless workgroups. A table marked as no-backup in an RG or RA3 cluster or serverless workgroup is treated as a permanent table that will always be backed up while taking a snapshot, and always restored when restoring from a snapshot. To avoid snapshot costs for no-backup tables, truncate them before taking a snapshot.

Restoring a snapshot to a serverless namespace replaces the current database with the database in the snapshot.

Restoring a snapshot to a serverless namespace is completed in two phases. The first phase completes in a few minutes, restores the data to your namespace, and makes it available for queries. The second phase of restoration is where your database is tuned, which can cause minor performance issues. This second phase can last from a few hours to several days, and in some cases, a couple of weeks. The amount of time depends on the size of the data, but performance progressively improves as the database gets tuned. At the end of this phase, your serverless namespace is fully tuned, and you can submit queries without performance issues.

To restore a snapshot to a serverless namespace

1. On the Amazon Redshift Serverless console, choose **Data backup**.
2. Choose the snapshot to restore. You can only restore one snapshot at a time.
3. Choose **Actions**, **Restore to serverless namespace**.
4. Choose an available namespace to restore to. You can only restore to namespaces whose statuses are Available.
5. Choose **Restore**.

To restore a snapshot to a provisioned cluster

1. On the Amazon Redshift Serverless console, choose **Data backup**.
2. Choose a snapshot to restore.
3. Choose **Action**, **Restore to provisioned cluster**.
4. Enter a cluster identifier.
5. Choose a **Node type**. The number of nodes depends on the node type.
6. Follow the instructions on the page on the console page to enter the properties for **Cluster configuration**. See Creating a cluster for more information.

For more information about snapshots on provisioned clusters, see Amazon Redshift snapshots and backups.

Converting a recovery point

Recovery points in Amazon Redshift Serverless are created approximately every 30 minutes and saved for 24 hours. To convert a recovery point to a snapshot, perform the steps in the following procedure.

To convert a recovery point to a snapshot

1. On the Amazon Redshift Serverless console, choose **Data backup**.
2. Under **Recovery points**, choose the **Creation time** of the recovery point that you want to convert to a snapshot.
3. Choose **Create snapshot from recovery point**.
4. Enter a **Snapshot identifier**.
5. Choose **Create**.

Restoring a recovery point

Recovery points in Amazon Redshift Serverless are created approximately every 30 minutes and saved for 24 hours. To restore a recovery point to a serverless namespace, perform the steps in the following procedure

To restore a recovery point to a serverless namespace

1. On the Amazon Redshift Serverless console, choose **Data backup**.
2. Under **Recovery points**, choose the **Creation time** of the recovery point that you want to restore.
3. Choose **Restore**. You can only restore to namespaces whose statuses are Available.
4. Enter **restore** in the text input field and choose **Restore**.

Copying backups to another AWS Region

You can configure Amazon Redshift Serverless to automatically copy snapshots and recovery points to another AWS Region. When you create a snapshot in the *source* AWS Region, it's copied to a *destination* Region. You can configure your namespace so that it only copies snapshots and recovery points to one destination AWS Region at a time. For a list of AWS Regions where Amazon

Redshift Serverless is available, see the endpoints listed for Redshift Serverless API in the *Amazon Web Services General Reference*.

When you configure copying backups, you can also specify a retention period of how long Amazon Redshift Serverless should keep the copied snapshot. You can't change the retention periods of recovery points, which must be 1 day. The retention periods of snapshots in the destination Region is separate from the retention period of the snapshot in the source Region. By default, the retention period is to keep the snapshot indefinitely. If you choose **Custom value** choose the number of days. This amount you choose must be between 1-3653 days, inclusive.

To change the destination Region to copy snapshots to, first disable copying backups, and then specify the new destination Region when you re-enable copying.

Once a snapshot or recovery point is copied to a destination Region, you can use it to restore data to the Region.

By default, your data is encrypted with a key that AWS manages for you. To use a different key, choose the key that you want to use when configuring backup copying in the source AWS Region, and Amazon Redshift Serverless automatically creates a grant, which enables snapshot encryption in the destination AWS Region.

To copy backups to another Region, make sure that you have the following IAM permissions:

```
redshift-serverless:CreateSnapshotCopyConfiguration
redshift-serverless:UpdateSnapshotCopyConfiguration
redshift-serverless:ListSnapshotCopyConfigurations
redshift-serverless:DeleteSnapshotCopyConfiguration
```

If you're using your own KMS key to encrypt your backups, you also need the following permissions:

```
kms:CreateGrant
kms:DescribeKey
```

To configure copying your snapshots or recovery points to another AWS Region

1. On the Amazon Redshift Serverless console, choose the namespace for which you want to configure copying snapshots or recovery points.
2. Choose the **Actions**, **Configure cross-Region backup**.
3. Choose the destination AWS Region to copy the snapshot to.

4. (Optional) Choose how long to retain the snapshot. If you choose **Custom value** choose the number of days The amount you choose must be between 1-3653 days, inclusive. The default is to retain indefinitely.
5. (Optional) Choose a different AWS KMS key to use to encrypt for encryption in the destination Region.
6. Choose **Save configuration**.

Restoring a table

You can also restore a specific table from a snapshot or a recovery point When you do so, you specify the source snapshot or recovery point, database, schema, table, the target database, schema, and new table name. This new table can't have the same name as an existing table. If you want to replace an existing table by restoring a table, you must first rename or drop the table before you restore the table.

> **Note**
>
> No-backup tables aren't supported for RG or RA3 provisioned clusters and Amazon Redshift Serverless workgroups. A table marked as no-backup in an RG or RA3 cluster or serverless workgroup is treated as a permanent table that will always be backed up while taking a snapshot, and always restored when restoring from a snapshot. However, selective restoration of no-backup tables isn't supported.

The target table is created using the source table's column definitions, table attributes, and column attributes except for foreign keys. To prevent conflicts due to dependencies, the target table doesn't inherit foreign keys from the source table. Any dependencies, such as views or permissions granted on the source table, aren't applied to the target table.

If the owner of the source table exists, then that user is the owner of the restored table, provided that the user has sufficient permissions to become the owner of a relation in the specified database and schema. Otherwise, the restored table is owned by the admin user that was created when the cluster was launched.

The restored table returns to the state it was in at the time the backup was taken. This includes transaction visibility rules defined by the Amazon Redshift adherence to serializable isolation, meaning that data will be immediately visible to in flight transactions started after the backup.

You can use the Amazon Redshift Serverless console to restore tables from a snapshot.

Restoring a table from data backup has the following limitations:

- You can only restore one table at a time.
- Any dependencies, such as views or permissions granted on the source table, aren't applied to the target table.
- If row-level security is turned on for a table being restored, Amazon Redshift Serverless restores the table with row-level security turned on.

To restore a table using the Amazon Redshift Serverless console

1. On the Amazon Redshift Serverless console, choose **Data backup**.
2. Choose the snapshot or recovery point that has the table to restore.
3. Choose **Actions**, **Restore table from snapshot** or **Restore table from recovery point**.
4. Enter information about the source snapshot or recovery point and target table, then choose **Restore table**.

Data sharing in Amazon Redshift Serverless

With *data sharing*, you have live access to data so that your users can see the most up-to-date and consistent information as it's updated in Amazon Redshift Serverless.

You can share data for read purposes across different Amazon Redshift Serverless instances within or across AWS accounts.

You can get started with data sharing by using either the SQL interface or the Amazon Redshift console. For more information, see Data sharing in Amazon Redshift in the *Amazon Redshift Database Developer Guide*.

With data sharing, Amazon Redshift Serverless namespaces and provisioned clusters can share live data with each other, whether they are within an AWS account across AWS accounts, or across AWS Regions. For more information, see Regions where data sharing is available.

To get started sharing data within an AWS account, open the AWS Management Console, and then choose the Amazon Redshift console. Choose **Namespace configuration** and then **Datashares**.

To start querying data in a datashare, create a database in a namespace that has a workgroup associated with it. From a specified datashare, choose a namespace that has a workgroup associated with it and create a database to query data.

Considerations

Consider the following when working with data sharing in Amazon Redshift Serverless:

- Amazon Redshift supports data sharing as producers or consumers on provisioned RG and RA3 clusters, and serverless endpoint workgroups.
- Amazon Redshift Serverless is encrypted by default.

For a list of datasharing limitations, including database objects supported, encryption requirements, and sort-key requirements, see Considerations for data sharing in Amazon Redshift in the *Amazon Redshift Database Developer Guide*.

Granting access to view datashares

A superuser can provide access to users who aren't superusers so that they can view the datashares created by all users.

To grant access to a datashare for a user, use the following command to provide datashare access for a user, where datashare_name is the name of the datashare and user-name is the name of the user for whom you want to provide access.

```
grant share on datashare datashare_name to "IAM:test_user";
```

To grant access to a datashare for a user group, first create a user group with users. For information on how to create user groups, see CREATE GROUP. Then, grant datashare access to a user using the following command, where datashare_name is the name of the datashare and user-group is the name of the user-group to that you want to grant access.

```
grant share on datashare datashare_name to group user_group;
```

For information on how to use the GRANT statement, see GRANT.

Registering namespaces to the AWS Glue Data Catalog

You can register entire namespaces to the AWS Glue Data Catalog and create catalogs managed by AWS Glue. You can access these catalogs with any SQL engine that supports the Apache Iceberg REST API. For more information on creating Apache Iceberg-compatible catalogs from Amazon Redshift see Apache Iceberg compatibility for Amazon Redshift in the Amazon Redshift Database Developer Guide.

To register a serverless namespace to the AWS Glue Data Catalog

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Redshift Serverless**. The Serverless dashboard appears. In the **Namespaces/Workgroups** section is the list of namespaces and workgroups for your account in the current AWS Region. If you don't have any namespaces, choose **Create workgroup** to create a workgroup and its corresponding namespace.
3. Choose the name of the namespace that you want to register.
4. From **Actions**, choose **Register to AWS Glue Data Catalog**. The **Register to AWS Glue Data Catalog** pop-up box appears.
5. Enter the AWS account ID that you want to register the namespace to under **Destination account ID**. This is the account ID that will hold the catalog in the AWS Glue Data Catalog.
6. Enter a name under **Register namespace as**. This will be the namespace's name in the Data Catalog.
7. Choose **Register**. You'll be taken to the AWS Lake Formation console.
8. Follow the catalog creation process in AWS Lake Formation. For information about creating a catalog, see Bringing Amazon Redshift data into the AWS Glue Data Catalog in the AWS Lake Formation Developer Guide.

Tagging resources in Amazon Redshift Serverless

In AWS, tags are user-defined labels that consist of key-value pairs. Amazon Redshift Serverless supports tagging to provide metadata about resources at a glance.

Tags are not required for resources, but they help provide context. You might want to tag resources with metadata with 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 for consistency.

If you delete a resource, any associated tags are deleted. You can use both the AWS CLI and Amazon Redshift Serverless console to tag serverless resources. Available API operations are `TagResource`, `UntagResource`, and `ListTagsForResource`.

Each resource has one tag set, which is a collection of one or more tags assigned to the resource. Each resource can have up to 50 tags per tag set. You can add tags when you create a resource and after a resource has been created. You can add tags to the following serverless resource types:

- Workgroups
- Namespaces
- Snapshots
- Recovery points

Tags have the following requirements:

- Keys can't be prefixed with `aws:`.
- Keys must be unique per tag set.
- A key must be between 1 and 128 allowed characters.
- A value must be between 0 and 256 allowed characters.
- Values do not need to be unique per tag set.
- Allowed characters for keys and values are Unicode letters, digits, white space, and any of the following symbols: _ . : / = + - @.
- Keys and values are case sensitive.

To manage tags of your Amazon Redshift Serverless resources

1. On the Amazon Redshift Serverless console, choose **Manage Tags**.
2. Enter the resource type to search for and choose **Search resources**. Choose the resource for which you want to manage tags, then choose **Manage tags**.
3. Specify the keys and optional values you want to add to the resource. When modifying a tag, you can change the tag's value, but not the key.

4. After you're done adding, removing, or modifying tags, choose **Save changes**, then choose **Apply** to save your changes.

Amazon Redshift provisioned clusters

An Amazon Redshift data warehouse is a collection of computing resources called *nodes*, which are organized into a group called a *cluster*. Each cluster runs an Amazon Redshift engine and contains one or more databases.

> **Note**
>
> At this time, Amazon Redshift version 1.0 engine is available. However, as the engine is updated, multiple Amazon Redshift engine versions might be available for selection.

Clusters and nodes in Amazon Redshift

An Amazon Redshift cluster consists of nodes. Each cluster has a leader node and one or more compute nodes. The *leader node* receives queries from client applications, parses the queries, and develops query execution plans. The leader node then coordinates the parallel execution of these plans with the compute nodes and aggregates the intermediate results from these nodes. It then finally returns the results back to the client applications.

Compute nodes run the query execution plans and transmit data among themselves to serve these queries. The intermediate results are sent to the leader node for aggregation before being sent back to the client applications. For more information about leader nodes and compute nodes, see Data warehouse system architecture in the *Amazon Redshift Database Developer Guide*.

> **Note**
>
> When you create a cluster on the Amazon Redshift console (https://console.aws.amazon.com/redshiftv2/), you can get a recommendation of your cluster configuration based on the size of your data and query characteristics. To use this sizing calculator, look for **Help me choose** on the console in AWS Regions that support RG or RA3 node types. For more information, see Creating a cluster.

When you launch a cluster, one option that you specify is the node type. The node type determines the CPU, RAM, storage capacity, and storage drive type for each node.

Amazon Redshift offers different node types to accommodate your workloads, and we recommend choosing RG or RA3 depending on the required performance, data size, and expected data growth.

RG and RA3 nodes with managed storage enable you to optimize your data warehouse by scaling and paying for compute and managed storage independently. With RG or RA3, you choose the number of nodes based on your performance requirements and only pay for the managed storage that you use. Size your RG or RA3 cluster based on the amount of data you process daily. You launch clusters that use the RG or RA3 node types in a virtual private cloud (VPC). For more information, see Creating a Redshift provisioned cluster or Amazon Redshift Serverless workgroup in a VPC.

Amazon Redshift managed storage uses large, high-performance SSDs in each RG and RA3 node for fast local storage and Amazon S3 for longer-term durable storage. If the data in a node grows beyond the size of the large local SSDs, Amazon Redshift managed storage automatically offloads that data to Amazon S3. You pay the same low rate for Amazon Redshift managed storage regardless of whether the data sits in high-performance SSDs or Amazon S3. For workloads that require ever-growing storage, managed storage lets you automatically scale your data warehouse storage capacity separate from compute nodes.

RG nodes provide the following features over RA3 node types:

- **Graviton-based instance types** – Deliver superior price performance for analytics workloads
- **Integrated data lake query engine** – RG clusters include an integrated data lake query engine that runs directly on the cluster's own compute resources. RA3 clusters use Amazon Redshift Spectrum for data lake queries.

DC2 nodes enable you to have compute-intensive data warehouses with local SSD storage included. You choose the number of nodes you need based on data size and performance requirements. DC2 nodes store your data locally for high performance, and as the data size grows, you can add more compute nodes to increase the storage capacity of the cluster. For datasets under 1 TB (compressed), we recommend DC2 node types for the best performance at the lowest price. If you expect your data to grow, we recommend using RA3 nodes so you can size compute and storage independently to achieve improved price and performance. You launch clusters that use the DC2 node types in a virtual private cloud (VPC). For more information, see Creating a Redshift provisioned cluster or Amazon Redshift Serverless workgroup in a VPC.

Node types are available in different sizes. Node size and the number of nodes determine the total storage for a cluster. For more information, see Node type details.

Some node types allow one node (single-node) or two or more nodes (multi-node). The minimum number of nodes for clusters of some node types is two nodes. On a single-node cluster, the node is shared for leader and compute functionality. Single-node clusters are not recommended for running production workloads. On a multi-node cluster, the leader node is separate from the compute nodes. The leader node is the same node type as the compute nodes. You only pay for compute nodes.

Amazon Redshift applies quotas to resources for each AWS account in each AWS Region. A *quota* restricts the number of resources that your account can create for a given resource type, such as nodes or snapshots, within an AWS Region. For more information about the default quotas that apply to Amazon Redshift resources, see Quotas and limits in Amazon Redshift.

The cost of your cluster depends on the AWS Region, node type, number of nodes, and whether the nodes are reserved in advance. For more information about the cost of nodes, see the Amazon Redshift pricing page.

Node type details

The following tables summarize the node specifications for each node type and size. The headings in the tables have these meanings:

- *vCPU* is the number of virtual CPUs for each node.
- *RAM* is the amount of memory in gibibytes (GiB) for each node.
- *Default slices per node* is the number of slices into which a compute node is partitioned when a cluster is created or resized with classic resize.

  The number of slices per node might change if the cluster is resized using elastic resize. However the total number of slices on all the compute nodes in the cluster remains the same after elastic resize.

  When you create a cluster with the restore from snapshot operation, the number of slices of the resulting cluster might change from the original cluster if you change the node type.
- *Storage* is the capacity and type of storage for each node.
- *Node range* is the minimum and maximum number of nodes that Amazon Redshift supports for the node type and size.

> **Note**
>
> You might be restricted to fewer nodes depending on the quota that is applied to your AWS account in the selected AWS Region. For more information about the default quotas that apply to Amazon Redshift resources, see Quotas and limits in Amazon Redshift.

- *Total capacity* is the total storage capacity for the cluster if you deploy the maximum number of nodes that is specified in the node range.

The following table describes specifications for RG nodes.

Node type	vCPU	RAM (GiB)	Default slices per node	Managed storage limit per node [1]	Node range with create cluster	Total managed storage capacity [2]
rg.large (single-node)	2	16	2	1 TB	1	1 TB
rg.large (multi-node)	2	16	2	8 TB	2–16	128 TB
rg.xlarge (multi-node)	4	32	2	32 TB	2–16[3]	1024 TB[3]
rg.4xlarge	16	128	8	128 TB	2–32[4]	8192 TB[4]
rg.12xlarge	48	384	16	128 TB	2–128	16,384 TB

[1] The storage limit for Amazon Redshift managed storage. This is a hard limit.

[2] Total managed storage limit is the maximum number of nodes times the managed storage limit per node.

[3] You can create a cluster with the rg.xlarge (multi-node) node type that has up to 16 nodes. For multiple-node clusters, you can resize with elastic resize to a maximum of 32 nodes.

[4] You can create a cluster with the rg.4xlarge (multi-node) node type with up to 32 nodes. You can resize it with elastic resize to a maximum of 64 nodes.

The following table describes specifications for RA3 nodes.

Node type	vCPU	RAM (GiB)	Default slices per node	Managed storage limit per node [1]	Node range with create cluster	Total managed storage capacity [2]
ra3.large (single-node)	2	16	2	1 TB	1	1 TB[3]
ra3.large (multi-node)	2	16	2	8 TB	2-16	128 TB
ra3.xlplus (single-node)	4	32	2	4 TB	1	4 TB[3]
ra3.xlplus (multi-node)	4	32	2	32 TB	2–16[4]	1024 TB[4]
ra3.4xlarge	12	96	4	128 TB	2–32[5]	8192 TB[5]
ra3.16xlarge	48	384	16	128 TB	2–128	16,384 TB

[1] The storage limit for Amazon Redshift managed storage. This is a hard limit.

[2] Total managed storage limit is the maximum number of nodes times the managed storage limit per node.

[3] To resize a single-node cluster to multi-node, only classic resize is supported.

[4] You can create a cluster with the ra3.xlplus (multi-node) node type that has up to 16 nodes. For multiple-node clusters, you can resize with elastic resize to a maximum of 32 nodes.

[5] You can create a cluster with the ra3.4xlarge node type with up to 32 nodes. You can resize it with elastic resize to a maximum of 64 nodes.

The following table describes specifications for dense compute nodes.

Node type	vCPU	RAM (GiB)	Default slices per node	Storage per node	Node range	Total capacity
dc2.large	2	15	2	160 GB NVMe-SSD	1–32	5.12 TB
dc2.8xlarge	32	244	16	2.56 TB NVMe-SSD	2–128	326 TB

Note

Dense storage (DS2) node types are no longer available.

Previous node type names

In previous releases of Amazon Redshift, certain node types had different names. You can use the previous names in the Amazon Redshift API and AWS CLI. However, we recommend that you update any scripts that reference those names to use the current names instead. The current and previous names are as follows.

Current name	Previous names
ds2.xlarge	ds1.xlarge, dw.hs1.xlarge, dw1.xlarge

Current name	Previous names
ds2.8xlarge	ds1.8xlarge, dw.hs1.8x large, dw1.8xlarge
dc1.large	dw2.large
dc1.8xlarge	dw2.8xlarge

Determining the number of nodes

Because Amazon Redshift distributes and runs queries in parallel across all of a cluster's compute nodes, you can increase query performance by adding nodes to your cluster. When you run a cluster with at least two compute nodes, data on each node is mirrored on disks of another node to reduce the risk of incurring data loss.

You can monitor query performance in the Amazon Redshift console and with Amazon CloudWatch metrics. You can also add or remove nodes as needed to achieve the balance between price and performance for your cluster. When you request an additional node, Amazon Redshift takes care of all the details of deployment, load balancing, and data maintenance. For more information about cluster performance, see Monitoring Amazon Redshift cluster performance.

Reserved nodes are appropriate for steady-state production workloads, and offer significant discounts over on-demand nodes. You can purchase reserved nodes after running experiments and proof-of-concepts to validate your production configuration. For more information, see Reserved nodes.

When you pause a cluster, you suspend on-demand billing during the time the cluster is paused. During this paused time, you only pay for backup storage. This frees you from planning and purchasing data warehouse capacity ahead of your needs, and enables you to cost-effectively manage environments for development or test purposes.

For information about pricing of on-demand and reserved nodes, see Amazon Redshift pricing.

Use EC2 to create your cluster

Amazon Redshift clusters run in Amazon EC2 instances that are configured for the Amazon Redshift node type and size that you select. For more information about these networking platforms, see Supported Platforms in the *Amazon EC2 User Guide*.

> **Note**
>
> To prevent connection issues between SQL client tools and the Amazon Redshift database, we recommend doing one of two things. You can configure an inbound rule that enables the hosts to negotiate packet size. Alternatively, you can disable TCP/IP jumbo frames by setting the maximum transmission unit (MTU) to 1500 on the network interface (NIC) of your Amazon EC2 instances. For more information about these approaches, see Queries appear to hang and sometimes fail to reach the cluster.

Amazon Virtual Private Cloud (Amazon VPC)

When using Amazon VPC, your cluster runs in a virtual private cloud (VPC) that is logically isolated to your AWS account. If you provision your cluster with Amazon VPC, you control access to your cluster by associating one or more VPC security groups with the cluster. For more information, see Security Groups for Your VPC in the *Amazon VPC User Guide*.

To create a cluster in a VPC, you must first create an Amazon Redshift cluster subnet group by providing subnet information of your VPC, and then provide the subnet group when launching the cluster. For more information, see Subnets for Redshift resources.

For more information about Amazon Virtual Private Cloud (Amazon VPC), see the Amazon VPC product detail page.

Default disk space alarm

When you create an Amazon Redshift cluster, you can optionally configure an Amazon CloudWatch alarm to monitor the average percentage of disk space that is used across all of the nodes in your cluster. We'll refer to this alarm as the *default disk space alarm*.

The purpose of default disk space alarm is to help you monitor the storage capacity of your cluster. You can configure this alarm based on the needs of your data warehouse. For example, you can use the warning as an indicator that you might need to resize your cluster. You might resize either to a different node type or to add nodes, or perhaps to purchase reserved nodes for future expansion.

The default disk space alarm triggers when disk usage reaches or exceeds a specified percentage for a certain number of times and at a specified duration. By default, this alarm triggers when the percentage that you specify is reached, and stays at or above that percentage for five minutes or longer. You can edit the default values after you launch the cluster.

When the CloudWatch alarm triggers, Amazon Simple Notification Service (Amazon SNS) sends a notification to specified recipients to warn them that the percentage threshold is reached. Amazon SNS uses a topic to specify the recipients and message that are sent in a notification. You can use an existing Amazon SNS topic; otherwise, a topic is created based on the settings that you specify when you launch the cluster. You can edit the topic for this alarm after you launch the cluster. For more information about creating Amazon SNS topics, see Getting Started with Amazon Simple Notification Service.

After you launch the cluster, you can view and edit the alarm from the cluster's **Status** window under **CloudWatch Alarms**. The name is **percentage-disk-space-used-default-<*string*>**. You can open the alarm to view the Amazon SNS topic that it is associated with and edit alarm settings. If you did not select an existing Amazon SNS topic to use, the one created for you is named **<*clustername*>-default-alarms (<*recipient*>)**; for example, **examplecluster-default-alarms (notify@example.com)**.

For more information about configuring and editing the default disk space alarm, see Creating a cluster and Creating a disk space alarm.

> **Note**
>
> If you delete your cluster, the alarm associated with the cluster will not be deleted but it will not trigger. You can delete the alarm from the CloudWatch console if you no longer need it.

Cluster status

The cluster status displays the current state of the cluster. The following table provides a description for each cluster status.

Status	Description
`available`	The cluster is running and available.
`available, prep-for-resize`	The cluster is being prepared for elastic resize. The cluster is running and available for read and write queries, but cluster operations, such as creating a snapshot, are not available.

Status	Description
`available, resize-cleanup`	An elastic resize operation is completing data transfer to the new cluster nodes. The cluster is running and available for read and write queries, but cluster operations, such as creating a snapshot, are not available.
`cancelling-resize`	The resize operation is being cancelled.
`creating`	Amazon Redshift is creating the cluster. For more information, see Creating a cluster.
`deleting`	Amazon Redshift is deleting the cluster. For more information, see Shutting down and deleting a cluster.
`final-snapshot`	Amazon Redshift is taking a final snapshot of the cluster before deleting it. For more information, see Shutting down and deleting a cluster.
`hardware-failure`	The cluster suffered a hardware failure. If you have a single-node cluster, the node cannot be replaced. To recover your cluster, restore a snapshot. For more information, see Amazon Redshift snapshots and backups.
`incompatible-hsm`	Amazon Redshift cannot connect to the hardware security module (HSM). Check the HSM configuration between the cluster and HSM. For more information, see Encryption using hardware security modules.
`incompatible-network`	There is an issue with the underlying network configuration. Make sure that the VPC in which you launched the cluster exists and its settings are correct. For more information, see Redshift resources in a VPC.
`incompatible-parameters`	There is an issue with one or more parameter values in the associated parameter group, and the parameter value or values cannot be applied. Modify the parameter group and update any invalid values. For more information, see Amazon Redshift parameter groups.

Status	Description
`incompatible-restore`	There was an issue restoring the cluster from the snapshot. Try restoring the cluster again with a different snapshot. For more information, see Amazon Redshift snapshots and backups.
`modifying`	Amazon Redshift is applying changes to the cluster. For more informati on, see Modifying a cluster.
`paused`	The cluster is paused. For more information, see Pausing and resuming a cluster.
`rebooting`	Amazon Redshift is rebooting the cluster. For more information, see Rebooting a cluster.
`renaming`	Amazon Redshift is applying a new name to the cluster. For more information, see Renaming a cluster.
`resizing`	Amazon Redshift is resizing the cluster. For more information, see Resizing a cluster.
`rotating-keys`	Amazon Redshift is rotating encryption keys for the cluster. For more information, see Encryption key rotation.
`storage-full`	The cluster has reached its storage capacity. Resize the cluster to add nodes or to choose a different node size. For more information, see Resizing a cluster.
`updating-hsm`	Amazon Redshift is updating the HSM configuration.

Considerations for using Amazon Redshift provisioned clusters

After your cluster is created, you can find information in this section about regions where features are available, maintenance tasks, node types, and usage limits.

Region and Availability Zone considerations

Amazon Redshift is available in several AWS Regions. By default, Amazon Redshift provisions your cluster in a randomly selected Availability Zone (AZ) within the AWS Region that you choose. All the cluster nodes are provisioned in the same Availability Zone.

You can optionally request a specific Availability Zone if Amazon Redshift is available in that zone. For example, if you already have an Amazon EC2 instance running in one Availability Zone, you might want to create your Amazon Redshift cluster in the same zone to reduce latency. On the other hand, you might want to choose another Availability Zone for higher availability. Amazon Redshift might not be available in all Availability Zones within an AWS Region.

For a list of supported AWS Regions where you can provision an Amazon Redshift cluster, see Amazon Redshift endpoints in the *Amazon Web Services General Reference*.

Cluster maintenance

Amazon Redshift periodically performs maintenance to apply upgrades to your cluster. During these updates, your Amazon Redshift cluster isn't available for normal operations. You have several ways to control how we maintain your cluster. For example, you can control when we deploy updates to your clusters. You can also choose whether your cluster runs the most recently released version, or the version released previously to the most recently released version. Finally, you have the option to defer non-mandatory maintenance updates for a period of time.

Maintenance windows

Amazon Redshift assigns a 30-minute maintenance window at random from an 8-hour block of time per AWS Region, occurring on a random day of the week (Monday through Sunday, inclusive).

Default maintenance windows

The following list shows the time blocks for each AWS Region from which the default maintenance windows are assigned:

- US East (N. Virginia) Region: 03:00–11:00 UTC
- US East (Ohio) Region: 03:00–11:00 UTC
- US West (N. California) Region: 06:00–14:00 UTC
- US West (Oregon) Region: 06:00–14:00 UTC

- Africa (Cape Town) Region: 20:00–04:00 UTC
- Asia Pacific (Hong Kong) Region: 13:00–21:00 UTC
- Asia Pacific (Hyderabad) Region: 16:30–00:30 UTC
- Asia Pacific (Jakarta) Region: 15:00–23:00 UTC
- Asia Pacific (Malaysia) Region: 14:00–22:00 UTC
- Asia Pacific (Melbourne) Region: 12:00–20:00 UTC
- Asia Pacific (Mumbai) Region: 16:30–00:30 UTC
- Asia Pacific (New Zealand) Region: 10:00–18:00 UTC
- Asia Pacific (Osaka) Region: 13:00–21:00 UTC
- Asia Pacific (Seoul) Region: 13:00–21:00 UTC
- Asia Pacific (Singapore) Region: 14:00–22:00 UTC
- Asia Pacific (Sydney) Region: 12:00–20:00 UTC
- Asia Pacific (Taipei) Region: 14:00–22:00 UTC
- Asia Pacific (Thailand) Region: 15:00–23:00 UTC
- Asia Pacific (Tokyo) Region: 13:00–21:00 UTC
- Canada (Central) Region: 03:00–11:00 UTC
- Canada West (Calgary) Region: 04:00–12:00 UTC
- China (Beijing) Region: 13:00–21:00 UTC
- China (Ningxia) Region: 13:00–21:00 UTC
- Europe (Frankfurt) Region: 06:00–14:00 UTC
- Europe (Ireland) Region: 22:00–06:00 UTC
- Europe (London) Region: 22:00–06:00 UTC
- Europe (Milan) Region: 21:00–05:00 UTC
- Europe (Paris) Region: 23:00–07:00 UTC
- Europe (Stockholm) Region: 23:00–07:00 UTC
- Europe (Zurich) Region: 20:00–04:00 UTC
- Israel (Tel Aviv) Region: 20:00–04:00 UTC
- Mexico (Central) Region: 04:00–12:00 UTC
- Europe (Spain) Region: 21:00–05:00 UTC
- Middle East (Bahrain) Region: 13:00–21:00 UTC

- Middle East (UAE) Region: 18:00–02:00 UTC
- South America (São Paulo) Region: 19:00–03:00 UTC

If a maintenance event is scheduled for a given week, it starts during the assigned 30-minute maintenance window. While Amazon Redshift is performing maintenance, it terminates any queries or other operations that are in progress. Downtime experienced during planned maintenance is not considered monthly downtime or unavailability within the Amazon Redshift Service Level Agreement. Most maintenance completes during the 30-minute maintenance window, but some maintenance tasks might continue running after the window closes. If there are no maintenance tasks to perform during the scheduled maintenance window, your cluster continues to operate normally until the next scheduled maintenance window.

You can change the scheduled maintenance window by modifying the cluster, either programmatically or by using the Amazon Redshift console. You can find the maintenance window and set the day and time it occurs for the cluster under the **Maintenance** tab.

It is possible for a cluster to restart outside of a maintenance window. There are a few reasons this can occur. One more common reason is that an issue has been detected with the cluster and maintenance operations are being performed to bring it back to a healthy state. For more information, see the article Why did my Amazon Redshift cluster reboot outside of the maintenance window?, which provides details regarding why this might occur.

Deferring maintenance

To reschedule your cluster's maintenance window, you can defer maintenance by up to 60 days. For example, if your cluster's maintenance window is set to Wednesday 08:30 – 09:00 UTC and you need to access your cluster at that time, you can defer the maintenance to a later time period.

If you defer maintenance, Amazon Redshift will still apply hardware updates or other mandatory security updates to your cluster. Your cluster isn't available during these updates.

If a hardware update or other mandatory security update is scheduled during the upcoming maintenance window, Amazon Redshift sends you advance notifications under the *Pending* category. To learn more about *Pending* event notifications, see Amazon Redshift provisioned cluster event notifications.

You can also choose to receive event notifications from Amazon Simple Notification Service (Amazon SNS). For more information about subscribing to event notifications from Amazon SNS, see Amazon Redshift cluster event notification subscriptions.

If you defer your cluster's maintenance, the maintenance window following the period of deferment can't be deferred.

> **Note**
>
> You can't defer maintenance after it has started.

For more information about cluster maintenance, see the following documentation:

- Maintenance windows
- Cluster operations
- Modifying a cluster

Choosing cluster maintenance tracks

When Amazon Redshift releases a new cluster version, your cluster is updated during its maintenance window. You can control whether your cluster is updated to the most recent release or to the previous release.

The track controls which cluster version is applied during a maintenance window. When Amazon Redshift releases a new cluster version, that version is assigned to the *current* track, and the previous version is assigned to the *trailing* track.

For information about cluster tracks, see Tracks for Amazon Redshift provisioned clusters and serverless workgroups.

Understanding how RG and RA3 nodes separate compute and storage

These sections detail tasks available for RG and RA3 node types, showing their applicability to a collection of use cases and detailing their advantages over previously available node types.

Advantages and availability of RG and RA3 nodes

RG and RA3 nodes provide the following advantages:

- They are flexible to grow your compute capacity without increasing your storage costs. And they scale your storage without over-provisioning compute capacity.
- They use high performance SSDs for your hot data and Amazon S3 for cold data. Thus they provide ease of use, cost-effective storage, and high query performance.

- They use high bandwidth networking built on the AWS Nitro System to further reduce the time taken for data to be offloaded to and retrieved from Amazon S3.

Consider choosing RG and RA3 node types in these cases:

- You need the flexibility to scale and pay for compute separate from storage.
- You query a fraction of your total data.
- Your data volume is growing rapidly or is expected to grow rapidly.
- You want the flexibility to size the cluster based only on your performance needs.

Data lake query compute considerations – On DC2 and RA3 provisioned clusters, data lake queries run on Redshift Spectrum, which uses dedicated Amazon Redshift servers that are independent of your cluster. On RG provisioned clusters, data lake queries run on the cluster's own compute resources and share those resources with other workloads. For workloads with heavy data lake query usage, consider this when sizing your RG cluster.

To use RG node types, your AWS Region must support RG. For more information, see RG node type availability in AWS Regions.

To use RA3 node types, your AWS Region must support RA3. For more information, see RA3 node type availability in AWS Regions.

Important

You can use ra3.xlplus node types only with cluster version 1.0.21262 or later. You can view the version of an existing cluster with the Amazon Redshift console. For more information, see Determining the workgroup or cluster version.
Make sure that you use the new Amazon Redshift console when working with RG or RA3 node types.
In addition, to use RG or RA3 node types with Amazon Redshift operations that use the track, the maintenance track value must be set to a cluster version that supports RG or RA3. For more information about tracks, see Choosing cluster maintenance tracks.

Consider the following when using single-node RA3 node types.

- Datasharing producers and consumers are supported.

- To change node types, only classic resize is supported. Changing the node type with elastic resize or snapshot restore isn't supported. The following scenarios are supported:
  - Classic resize of a 1-node dc2.xlarge to a 1-node ra3.xlplus, and vice versa.
  - Classic resize of a 1-node dc2.xlarge to a multiple-node ra3.xlplus, and vice versa.
  - Classic resize of a multiple-node dc2.xlarge to a 1-node ra3.xlplus, and vice versa.

Working with Amazon Redshift managed storage

With Amazon Redshift managed storage, you can store and process all your data in Amazon Redshift while getting more flexibility to scale compute and storage capacity separately. You continue to ingest data with the COPY or INSERT command. To optimize performance and manage automatic data placement across tiers of storage, Amazon Redshift takes advantage of optimizations such as data block temperature, data block age, and workload patterns. When needed, Amazon Redshift scales storage automatically to Amazon S3 without requiring any manual action.

For information about storage costs, see Amazon Redshift pricing.

Managing RG or RA3 node types

To take advantage of separating compute from storage, you can create or upgrade your cluster with the RG or RA3 node type. To use the RG or RA3 node types, create your clusters in a virtual private cloud (EC2-VPC).

To change the number of nodes of Amazon Redshift cluster with an RG or RA3 node type, do one of the following:

- Add or remove nodes with the elastic resize operation. In some situations, removing nodes from a RG or RA3 cluster isn't allowed with elastic resize. For example, when a 2:1 node count upgrade puts the number of slices per node at 32. For more information, see Resizing a cluster. If elastic resize isn't available, use classic resize.
- Add or remove nodes with the classic resize operation. Choose this option when you are resizing to a configuration that isn't available through elastic resize. Elastic resize is quicker than classic resize. For more information, see Resizing a cluster.

RG node type availability in AWS Regions

The RG node types are available only in the following AWS Regions:

- US East (N. Virginia) Region (us-east-1)
- US East (Ohio) Region (us-east-2)
- US West (N. California) Region (us-west-1)
- US West (Oregon) Region (us-west-2)
- Africa (Cape Town) Region (af-south-1)
- Asia Pacific (Hong Kong) Region (ap-east-1)
- Asia Pacific (Taipei) Region (ap-east-2)
- Asia Pacific (Tokyo) Region (ap-northeast-1)
- Asia Pacific (Seoul) Region (ap-northeast-2)
- Asia Pacific (Osaka) Region (ap-northeast-3)
- Asia Pacific (Mumbai) Region (ap-south-1)
- Asia Pacific (Hyderabad) Region (ap-south-2)
- Asia Pacific (Singapore) Region (ap-southeast-1)
- Asia Pacific (Sydney) Region (ap-southeast-2)
- Asia Pacific (Jakarta) Region (ap-southeast-3)
- Asia Pacific (Melbourne) Region (ap-southeast-4)
- Asia Pacific (Malaysia) Region (ap-southeast-5)
- Asia Pacific (Thailand) Region (ap-southeast-7)
- Canada (Central) Region (ca-central-1)
- Europe (Frankfurt) Region (eu-central-1)
- Europe (Zurich) Region (eu-central-2)
- Europe (Stockholm) Region (eu-north-1)
- Europe (Milan) Region (eu-south-1) (rg.xlarge and rg.4xlarge only)
- Europe (Spain) Region (eu-south-2)
- Europe (Ireland) Region (eu-west-1)
- Europe (London) Region (eu-west-2)
- Europe (Paris) Region (eu-west-3)
- Mexico (Central) Region (mx-central-1)
- South America (São Paulo) Region (sa-east-1)
- AWS GovCloud (US-East) (us-gov-east-1)

- AWS GovCloud (US-West) (us-gov-west-1)

RA3 node type availability in AWS Regions

The RA3 node types are available only in the following AWS Regions:

- US East (N. Virginia) Region (us-east-1)
- US East (Ohio) Region (us-east-2)
- US West (N. California) Region (us-west-1)
- US West (Oregon) Region (us-west-2)
- Africa (Cape Town) Region (af-south-1)
- Asia Pacific (Hong Kong) Region (ap-east-1)
- Asia Pacific (Hyderabad) Region (ap-south-2)
- Asia Pacific (Jakarta) Region (ap-southeast-3)
- Asia Pacific (Malaysia) Region (ap-southeast-5)
- Asia Pacific (Melbourne) Region (ap-southeast-4)
- Asia Pacific (Mumbai) Region (ap-south-1)
- Asia Pacific (New Zealand) Region (ap-southeast-6)
- Asia Pacific (Osaka) Region (ap-northeast-3)
- Asia Pacific (Seoul) Region (ap-northeast-2)
- Asia Pacific (Singapore) Region (ap-southeast-1)
- Asia Pacific (Sydney) Region (ap-southeast-2)
- Asia Pacific (Taipei) Region (ap-east-2)
- Asia Pacific (Thailand) Region (ap-southeast-7)
- Asia Pacific (Tokyo) Region (ap-northeast-1)
- Canada (Central) Region (ca-central-1)
- Canada West (Calgary) Region (ca-west-1)
- China (Beijing) Region (cn-north-1)
- China (Ningxia) Region (cn-northwest-1)
- Europe (Frankfurt) Region (eu-central-1)
- Europe (Zurich) Region (eu-central-2)

- Europe (Ireland) Region (eu-west-1)
- Europe (London) Region (eu-west-2)
- Europe (Milan) Region (eu-south-1)
- Europe (Spain) Region (eu-south-2)
- Europe (Paris) Region (eu-west-3)
- Europe (Stockholm) Region (eu-north-1)
- Israel (Tel Aviv) Region (il-central-1)
- Mexico (Central) Region (mx-central-1)
- Middle East (Bahrain) Region (me-south-1)
- Middle East (UAE) Region (me-central-1)
- South America (São Paulo) Region (sa-east-1)
- AWS GovCloud (US-East) (us-gov-east-1)
- AWS GovCloud (US-West) (us-gov-west-1)

Upgrading to RG or RA3 node types

To upgrade your existing node type to RG or RA3, you have the following options to change the node type:

- Restore from a snapshot – Amazon Redshift uses the most recent snapshot of your cluster and restores it to create a new RG or RA3 cluster. As soon as the cluster creation is complete (usually within minutes), RG or RA3 nodes are ready to run your full production workload. As compute is separate from storage, hot data is brought in to the local cache at fast speeds thanks to a large networking bandwidth. If you restore from the latest DC2 snapshot, RG and RA3 preserves hot block information of the DC2 workload and populates its local cache with the hottest blocks. For more information, see Restoring a cluster from a snapshot.

  To keep the same endpoint for your applications and users, you can rename the new RG or RA3 cluster with the same name as the original DC2 cluster. To rename the cluster, modify the cluster in the Amazon Redshift console or `ModifyCluster` API operation. For more information, see Renaming a cluster or `ModifyCluster` API operation in the *Amazon Redshift API Reference*.
- Elastic resize – resize the cluster using elastic resize. When you use elastic resize to change node type, Amazon Redshift automatically creates a snapshot, creates a new cluster, deletes the old cluster, and renames the new cluster. The elastic resize operation can be run on-demand or

can be scheduled to run at a future time. You can quickly upgrade your existing DC2 node type clusters to RG or RA3 with elastic resize. For more information, see Elastic resize.

> **Note**
>
> To migrate existing RA3 clusters to rg.xlarge or rg.4xlarge, your source cluster must run patch version P201 or later. To migrate to rg.large or rg.12xlarge, your source cluster must run P202 or later. To create or migrate to a single-node rg.large cluster, your cluster must run P204 or later. DC2 clusters can migrate to RG regardless of patch version.

The following table shows recommendations when upgrading to RG node types.

The recommendations in this table are starting cluster node types and sizes but depend on the computing requirements of your workload. RG has different performance characteristics, so the recommended mapping is a starting point. Benchmark your critical workloads and adjust the number of nodes up or down based on observed performance. RG instances are lower in price per vCPU compared to RA3, so adding nodes to optimize performance can still result in lower overall cost. For more information about node type pricing, see Amazon Redshift pricing.

To better estimate your requirements, consider conducting a proof of concept (POC) that uses Test Drive to run potential configurations. Provision a cluster for your POC data warehouse instead of Redshift Serverless. For more information about conducting a proof of concept, see Conduct a proof of concept (POC) for Amazon Redshift in the *Amazon Redshift Database Developer Guide*.

Existing node type	Existing number of nodes	Recommended new node type	Upgrade action
ra3.16xlarge	2–128	rg.12xlarge	Start with 1 node of rg.12xlarge for every 1 node of ra3.16xlarge[1].
ra3.4xlarge	2–64	rg.4xlarge	Start with 3 nodes of rg.4xlarge for every 4 nodes of ra3.4xlarge[1].

Existing node type	Existing number of nodes	Recommended new node type	Upgrade action
ra3.xlplus	1	rg.large	Start with 2 nodes of rg.large for a single-node ra3.xlplus cluster[1,2].
ra3.xlplus	2–32	rg.xlarge	Start with 1 node of rg.xlarge for every 1 node of ra3.xlplus[1].
ra3.large	1–16	rg.large	Start with 1 node of rg.large for every 1 node of ra3.large[1,2].
dc2.8xlarge	2–15	rg.4xlarge	Start with 3 nodes of rg.4xlarge for every 2 nodes of dc2.8xlarge[1].
dc2.8xlarge	16–128	rg.12xlarge	Start with 1 node of rg.12xlarge for every 2 nodes of dc2.8xlarge[1].
dc2.large	1–3	rg.large	Start with 1 node of rg.large for every 1 node of dc2.large[1,2].
dc2.large	4	rg.large	Start with 3 nodes of rg.large[1].
dc2.large	5–15	rg.xlarge	Start with 3 nodes of rg.xlarge for every 8 nodes of dc2.large[1].

Existing node type	Existing number of nodes	Recommended new node type	Upgrade action
dc2.large	16–32	rg.4xlarge	Start with 1 node of rg.4xlarge for every 10 nodes of dc2.large [1].

[1]Extra nodes might be needed depending on workload requirements. Add or remove nodes based on the compute requirements of your required query performance. RG instances are lower in price per vCPU compared to RA3, so adding nodes can still result in lower overall cost compared to your previous configuration. For more information, see Amazon Redshift pricing.

[2]For single-node clusters, see **Single-node clusters** under Things to consider when upgrading to RG nodes.

The following table shows recommendations when upgrading to RA3 node types.

The recommendations in this table are starting cluster node types and sizes but depend on the computing requirements of your workload. To better estimate your requirements, consider conducting a proof of concept (POC) that uses Test Drive to run potential configurations. Provision a cluster for your POC data warehouse instead of Redshift Serverless. For more information about conducting a proof of concept, see Conduct a proof of concept (POC) for Amazon Redshift in the *Amazon Redshift Database Developer Guide*.

Existing node type	Existing number of nodes	Recommended new node type	Upgrade action
dc2.8xlarge	2–15	ra3.4xlarge	Start with 2 nodes of ra3.4xlarge for every 1 node of dc2.8xlarge[1].
dc2.8xlarge	16–128	ra3.16xlarge	Start with 1 node of ra3.16xlarge for every 2 nodes of dc2.8xlarge[1].

Existing node type	Existing number of nodes	Recommended new node type	Upgrade action
dc2.large	1–4	ra3.large	Start with 1 node of ra3.large for every 1 node of dc2.large[1]. Start with 2 nodes of ra3.large for every 2 nodes of dc2.large[1]. Start with 3 nodes of ra3.large for every 3 nodes of dc2.large[1]. Start with 3 nodes of ra3.large for every 4 nodes of dc2.large[1].
dc2.large	5–15	ra3.xlplus	Start with 3 nodes of ra3.xlplus for every 8 nodes of dc2.large[1].
dc2.large	16–32	ra3.4xlarge	Start with 1 node of ra3.4xlarge for every 8 nodes of dc2.large[1,2].

[1]Extra nodes might be needed depending on workload requirements. Add or remove nodes based on the compute requirements of your required query performance.

[2]Clusters with the dc2.large node type are limited to 32 nodes.

The minimum number of nodes for some RA3 node types is 2 nodes. Take this into consideration when creating an RA3 cluster.

Networking features supported by RG and RA3 nodes

RG and RA3 nodes support a collection of networking features not available to other node types. This section provides brief descriptions of each feature and links to additional documentation:

- **Provisioned-cluster VPC endpoint** – When you create or restore an RG or RA3 cluster, Amazon Redshift uses a port within the ranges of 5431-5455 or 8191-8215. When the cluster is set to a port in one of these ranges, Amazon Redshift automatically creates a VPC endpoint in your AWS account for the cluster and attaches a private IP address to it. If you set the cluster to publicly-accessible, Redshift creates an elastic IP address in your AWS account and attaches it to the VPC endpoint. For more information, see Configuring security group communication settings for an Amazon Redshift cluster or an Amazon Redshift Serverless workgroup.
- **Single-subnet RG or RA3 clusters** – You can create an RG or RA3 cluster with a single subnet, but it can't use disaster-recovery features. An exception occurs if you enable cluster relocation when the subnet doesn't have multiple Availability Zones (AZs).
- **Multi-subnet RG or RA3 clusters and subnet groups** – You can create an RG or RA3 cluster with multiple subnets by creating a subnet group when you provision the cluster in your virtual private cloud (VPC). A cluster subnet group allows you to specify a set of subnets in your VPC and Amazon Redshift creates the cluster in one of them. After creating a subnet group, you can remove subnets you previously added, or add more. For more information, see Amazon Redshift cluster subnet groups.
- **Cross-account or cross-VPC endpoint access** – You can access a provisioned cluster or Amazon Redshift Serverless workgroup by setting up a Redshift-managed VPC endpoint. You can set it up as a private connection between a VPC that contains a cluster or workgroup and a VPC where you run a client tool, for example. By doing this, you can access the data warehouse without using a public IP address and without routing traffic through the internet. For more information, see Working with Redshift-managed VPC endpoints.
- **Cluster relocation** – You can move a cluster to another Availability Zone (AZ) without any loss of data when there is an interruption of service. You enable it on the console. For more information, see Relocating a cluster.
- **Custom domain name** – You can create a custom domain name, also known as a custom URL, for your Amazon Redshift cluster. It's an easy-to-read DNS record that routes SQL-client connections to your cluster endpoint. For more information, see Custom domain names for client connections.

Things to consider when upgrading to RG nodes

Patch Version

To migrate from RA3 to RG, your source cluster must meet the minimum required patch version: P201 for rg.xlarge and rg.4xlarge, P202 for multi-node rg.large and rg.12xlarge and P204 for single-node rg.large. DC2 clusters can migrate to RG regardless of patch version. Before migrating, test your workloads by restoring a snapshot to an RG cluster before moving production traffic. For details, see Amazon Redshift patch 201, Amazon Redshift patch 202 and Amazon Redshift patch 204.

Single-node clusters

- Single-node RG clusters are supported on rg.large only.
- A single-node rg.large cluster supports up to 1 TB of managed storage, compared to up to 4 TB for a single-node ra3.xlplus cluster. Use a multi-node rg.large cluster if your data is larger than 1 TB or is expected to grow beyond it.
- Single-node clusters are intended for development and test workloads. For production workloads, use at least two nodes. For more information, see Clusters and nodes in Amazon Redshift and Node type details.

Spectrum workloads

Clusters with workloads that heavily use Spectrum may observe slower query performance after migrating to RG. Unlike RA3 and DC2, RG runs the integrated data lake query engine directly on cluster compute nodes rather than on a separate Spectrum fleet. If you experience degraded Spectrum performance, increase the number of nodes in your RG cluster or migrate to a larger RG node type.

RG clusters also support querying data in Amazon S3 buckets located in a different AWS Region, removing the same-Region limitation that applies to RA3 and DC2 clusters using Spectrum. Cross-region queries incur additional data transfer charges. For more information, see Querying your data lake.

Cursor-based workloads

RG nodes have different supported cursor size compared to RA3 and DC2. Refer to Cursor Constraints for cursor size limits. If your workload relies heavily on cursors and you encounter

cursor size limitations, migrate to a larger RG node type — you can decrease the number of nodes proportionally to maintain a similar total cluster size and cost.

Concurrency Scaling eligibility

If your cluster uses Concurrency Scaling and has more than 32 nodes, and you want to migrate to rg.12xlarge, use elastic resize or snapshot restore to maintain Concurrency Scaling support. If your cluster has more than 32 nodes and you perform a classic resize to rg.12xlarge, the cluster loses Concurrency Scaling support. Freshly created rg.12xlarge clusters with more than 32 nodes do not support Concurrency Scaling. For more information, see Concurrency scaling candidates.

Reserved nodes

If your cluster uses reserved nodes, changing the node type doesn't automatically change your reservations – the existing reservations remain tied to the original node type until their term ends. Before you migrate a cluster that uses reserved nodes, see Things to consider before changing node type on a cluster that uses Reserved Instances.

Cluster operations

After you create a cluster, you can perform cluster operations to optimize performance, control costs, and ensure high availability. Cluster operations allow you to resize, pause, resume, or even recreate clusters as your data warehousing needs evolve.

Common use cases include scaling compute capacity for peak workloads, pausing clusters during inactive periods to reduce costs, and recreating clusters with different configurations or in different Availability Zones for disaster recovery. The following sections cover the details of performing various cluster operations to effectively manage your Amazon Redshift environment.

Creating a cluster

With Amazon Redshift, you can create a provisioned cluster to launch a new data warehouse. A provisioned cluster is a collection of computing resources called nodes, which are organized into a single, massively parallel processing (MPP) system.

Before you create a cluster, read Amazon Redshift provisioned clusters and Clusters and nodes in Amazon Redshift.

To create a cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**. The clusters for your account in the current AWS Region are listed. A subset of properties of each cluster is displayed in columns in the list.
3. Choose **Create cluster** to create a cluster.
4. Follow the instructions on the console page to enter the properties for **Cluster configuration**.

   The following step describes an Amazon Redshift console that is running in an AWS Region that supports RG or RA3 node types. For a list of AWS Regions that support RG or RA3 node types, see RG node type availability in AWS Regions and RA3 node type availability in AWS Regions in the *Amazon Redshift Management Guide*.

   If you don't know how large to size your cluster, choose **Help me choose**. Doing this starts a sizing calculator that asks you questions about the size and query characteristics of the data that you plan to store in your data warehouse. If you know the required size of your cluster (that is, the node type and number of nodes), choose **I'll choose**. Then choose the **Node type** and number of **Nodes** to size your cluster for the proof of concept.

   > **Note**
   >
   > If your organization is eligible and your cluster is being created in an AWS Region where Amazon Redshift Serverless is unavailable, you might be able to create a cluster under the Amazon Redshift free trial program. Choose either **Production** or **Free trial** to answer the question **What are you planning to use this cluster for?** When you choose **Free trial**, you create a configuration with the ra3.large node type. For more information about choosing a free trial, see Amazon Redshift free trial. For a list of AWS Regions where Amazon Redshift Serverless is available, see the endpoints listed for the Redshift Serverless API in the *Amazon Web Services General Reference*.

5. In the **Database configuration** section, specify a value for **Admin user name**. For **Admin password**, you can choose from the following options:

   - **Generate a password** – Use a password generated by Amazon Redshift.
   - **Manually add an admin password** – Use your own password.

- **Manage admin credentials in AWS Secrets Manager** – Amazon Redshift uses AWS Secrets Manager to generate and manage your admin password. Using AWS Secrets Manager to generate and manage your password's secret incurs a fee. For information on AWS Secrets Manager pricing, see AWS Secrets Manager Pricing.

6. (Optional) Follow the instructions on the console page to enter properties for **Cluster permissions**. Provide cluster permissions if your cluster needs to access other AWS services for you, for example to load data from Amazon S3.
7. Choose **Create cluster** to create the cluster. The cluster might take several minutes to be ready to use.

Additional configurations

When you create a cluster, you can specify additional properties to customize it. You can find more details about some of these properties in the following list.

IP address type

Choose the IP address type for your cluster. You can choose to have your resources communicate only over the IPv4 addressing protocol, or choose dual-stack mode, which lets your resources communicate over both IPv4 and IPv6. This feature is only available in the AWS GovCloud (US-East) and AWS GovCloud (US-West) Regions. For more information on AWS Regions, see Regions and Availability Zones.

Virtual private cloud (VPC)

Choose a VPC that has a cluster subnet group. After the cluster is created, the cluster subnet group can't be changed.

Parameter groups

Choose a cluster parameter group to associate with the cluster. If you don't choose one, the cluster uses the default parameter group.

Encryption

Choose whether you want to encrypt all data within the cluster and its snapshots. If you leave the default setting, **None**, encryption is not enabled. If you want to enable encryption, choose whether you want to use AWS Key Management Service (AWS KMS) or a hardware security module (HSM), and then configure the related settings. For more information about encryption in Amazon Redshift, see Amazon Redshift database encryption.

- **KMS**

  Choose **Use AWS Key Management Service (AWS KMS)** if you want to enable encryption and use AWS KMS to manage your encryption key. Also, choose the key to use. You can choose a default key, a key from the current account, or a key from a different account.

  > **Note**
  >
  > If you want to use a key from another AWS account, then enter the Amazon Resource Name (ARN) for the key to use. You must have permission to use the key. For more information about access to keys in AWS KMS, see Controlling access to your keys in the *AWS Key Management Service Developer Guide*.

  For more information about using AWS KMS encryption keys in Amazon Redshift, see Encryption using AWS KMS.

- **HSM**

  Choose **HSM** if you want to enable encryption and use a hardware security module (HSM) to manage your encryption key.

  If you choose **HSM**, choose values from **HSM Connection** and **HSM Client Certificate**. These values are required for Amazon Redshift and the HSM to form a trusted connection over which the cluster key can be passed. The HSM connection and client certificate must be set up in Amazon Redshift before you launch a cluster. For more information about setting up HSM connections and client certificates, see Encryption using hardware security modules.

Maintenance track

You can choose whether the cluster version used is the **Current**, **Trailing**, or sometimes **Preview** track.

Monitoring

You can choose whether to create CloudWatch alarms.

Configure cross-region snapshot

You can choose whether to enable cross-Region snapshots.

Automated snapshot retention period

You can choose the number of days to retain these snapshots within 35 days. If the node type is DC2, you can choose zero (0) days to not create automated snapshots.

Manual snapshot retention period

You can choose the number of days or `Indefinitely` to retain these snapshots.

Extra compute resources for automatic optimizations

You can choose whether to allocate extra compute resources to perform automatic optimizations, even during periods of heavy usage. For more information, see Allocating extra compute resources for automatic database optimization in the *Amazon Redshift Database Developer Guide*.

Creating a disk space alarm

You can monitor disk space usage and set alarms to be notified when disk space exceeds a specified threshold for a cluster. Creating a disk space usage alarm allows you to proactively manage storage capacity and prevent issues caused by insufficient disk space, such as query failures or data ingestion errors. The following procedure guides you through the process of creating a disk space usage alarm.

To create a disk space usage alarm for a cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Alarms**.
3. For **Actions**, choose **Create alarm**. The **Create alarm** page appears.
4. Follow the instructions on the page.
5. Choose **Create alarm**.

Viewing a cluster

Viewing a cluster allows you to monitor and manage your cluster's configuration, status, and performance metrics. By viewing cluster details, you can gain insights into resource utilization, query execution times, and system health. The following procedure shows you how to access cluster information.

To view a cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**. The clusters for your account in the current AWS Region are listed. A subset of properties of each cluster is displayed in columns in the list. If you don't have any clusters, choose **Create cluster** to create one.
3. Choose the cluster name in the list to view more details about a cluster.

Modifying a cluster

When you modify a cluster, changes to the following options are applied immediately:

- **VPC security groups**
- **Publicly accessible**
- **Admin user password**
- **HSM Connection**
- **HSM Client Certificate**
- **Maintenance detail**
- **Snapshot preferences**

Changes to the following options take effect only after the cluster is restarted:

- **Cluster identifier**

  Amazon Redshift restarts the cluster automatically when you change **Cluster identifier**.
- **Enhanced VPC routing**

  Amazon Redshift restarts the cluster automatically when you change **Enhanced VPC routing**.
- **Cluster parameter group**
- **IP address type**

  This feature is only available in the AWS GovCloud (US-East) and AWS GovCloud (US-West) Regions. For more information on AWS Regions, see Regions and Availability Zones.

If you decrease the automated snapshot retention period, existing automated snapshots whose settings fall outside of the new retention period are deleted. For more information, see Amazon Redshift snapshots and backups.

For more information about cluster properties, see Additional configurations.

To modify a cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**.
3. Choose the cluster to modify.
4. Choose **Edit**. The **Edit cluster** page appears.
5. Update the cluster properties. Some of the properties you can modify are:

   - Cluster identifier
   - Snapshot retention
   - Cluster relocation

   To edit settings for **Network and security**, **Maintenance**, and **Database configurations**, the console provides links to the appropriate cluster details tab.
6. Choose **Save changes**.

Resizing a cluster

As your data warehousing capacity and performance needs change, you can resize your cluster to make the best use of Amazon Redshift's computing and storage options.

When you resize a cluster, you specify a number of nodes or node type that is different from the current configuration of the cluster. While the cluster is in the process of resizing, you cannot run any write or read/write queries on the cluster; you can run only read queries.

For more information about resizing clusters, including walking through the process of resizing clusters using different approaches, see Resizing a cluster.

To resize a cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**.
3. Choose the cluster to resize.
4. For **Actions**, choose **Resize**. The **Resize cluster** page appears.
5. Follow the instructions on the page. You can resize the cluster now, once at a specific time, or increase and decrease the size of your cluster on a schedule.
6. Depending on your choices, choose **Resize now** or **Schedule resize**.

If you have reserved nodes, you can upgrade to RG or RA3 reserved nodes. You can do this when you use the console to restore from a snapshot or to perform an elastic resize. You can use the console to guide you through this process. For more information about upgrading to RG or RA3 node types, see Upgrading to RG or RA3 node types.

When you perform a resize operation to upgrade from a DC2.large node type to an RA3.large node type, Amazon Redshift automatically converts interleaved sort keys to compound sort keys. This conversion enables access to the concurrency scaling feature, which does not support queries on tables with interleaved sort keys. While this automatic conversion ensures compatibility with RA3 features, it may impact your existing query performance patterns.

If you want to maintain interleaved sort keys after upgrading to RA3 nodes, you can recreate your tables with the desired sort key configuration after the resize operation completes. However, choosing this option means you won't be able to use concurrency scaling for these tables.

A resize operation comes in two types:

- **Elastic resize** – You can add nodes to or remove nodes from your cluster. You can also change the node type, such as from DC2 nodes to RG or RA3 nodes. An elastic resize typically completes quickly, taking ten minutes on average. For this reason, we recommend it as a first option. When you perform an elastic resize, it redistributes data slices, which are partitions that are allocated memory and disk space in each node. Elastic resize is appropriate when you:
  - *Add or reduce nodes in an existing cluster, but you don't change the node type* – This is commonly called an *in-place* resize. When you perform this type of resize, some running queries complete successfully, but others can be dropped as part of the operation.

- *Change the node type for a cluster* – When you change the node type, a snapshot is created and data is redistributed from the source cluster to a cluster comprised of the new node type. On completion, running queries are dropped. Like the *in-place* resize, it completes quickly.
- **Classic resize** – You can change the node type, number of nodes, or both, in a similar manner to elastic resize. Classic resize takes more time to complete, but it can be useful in cases where the change in node count or the node type to migrate to doesn't fall within the bounds for elastic resize. This can apply, for instance, when the change in node count is really large.

Topics

- Elastic resize
- Classic resize

Elastic resize

An elastic resize operation, when you add or remove nodes of the same type, has the following stages:

1. Elastic resize takes a cluster snapshot. No-backup tables are only supported for DC2 nodes. For RG and RA3 clusters no-backup tables are included in the snapshot. For more information, see Excluding tables from snapshots. If your cluster doesn't have a recent snapshot, because you disabled automated snapshots, the backup operation can take longer. (To minimize the time before the resize operation begins, we recommend that you enable automated snapshots or create a manual snapshot before starting the resize.) When you start an elastic resize and a snapshot operation is in progress, the resize can fail if the snapshot operation doesn't complete within a few minutes. For more information, see Amazon Redshift snapshots and backups.
2. The operation migrates cluster metadata. The cluster is unavailable for a few minutes. The majority of queries are temporarily paused and connections are held open. It is possible, however, for some queries to be dropped. This stage is short.
3. Session connections are reinstated and queries resume.
4. Elastic resize redistributes data to node slices, in the background. The cluster is available for read and write operations, but some queries can take longer to run.
5. After the operation completes, Amazon Redshift sends an event notification.

When you use elastic resize to change the node type, it works similarly to when you add or subtract nodes of the same type. First, a snapshot is created. A new target cluster is provisioned with the

latest data from the snapshot, and data is transferred to the new cluster in the background. During this period, data is read only. When the resize nears completion, Amazon Redshift updates the endpoint to point to the new cluster and all connections to the source cluster are dropped.

It's unlikely that an elastic resize would fail. However, in the case of a failure, rollback happens automatically in the majority of cases without needing any manual intervention.

If you have reserved nodes, for example DC2 reserved nodes, you can upgrade to RA3 reserved nodes when you perform a resize. You can do this when you perform an elastic resize or use the console to restore from a snapshot. The console guides you through this process. For more information about upgrading to RG or RA3 node types, see Upgrading to RG or RA3 node types.

Elastic resize doesn't sort tables or reclaims disk space, so it isn't a substitute for a vacuum operation. For more information, see Vacuuming tables.

Elastic resize has the following constraints:

- *Elastic resize and data sharing clusters* - When you add or subtract nodes on a cluster that's a producer for data sharing, you can't connect to it from consumers while Amazon Redshift migrates cluster metadata. Similarly, if you perform an elastic resize and choose a new node type, data sharing is unavailable while connections are dropped and transferred to the new target cluster. In both types of elastic resize, the producer is unavailable for several minutes.
- *Data transfer from a shared snapshot* - To run an elastic resize on a cluster that is transferring data from a shared snapshot, at least one backup must be available for the cluster. You can view your backups on the Amazon Redshift console snapshots list, the `describe-cluster-snapshots` CLI command, or the `DescribeClusterSnapshots` API operation.
- *Platform restriction* - Elastic resize is available only for clusters that use the EC2-VPC platform. For more information, see Use EC2 to create your cluster.
- *Storage considerations* - Make sure that your new node configuration has enough storage for existing data. You may have to add additional nodes or change configuration.
- *Source vs target cluster size* - The number of nodes and node type that it's possible to resize to with elastic resize is determined by the number of nodes in the source cluster and the node type chosen for the resized cluster. To determine the possible configurations available, you can use the console. Or you can use the `describe-node-configuration-options` AWS CLI command with the `action-type resize-cluster` option. For more information about the resizing using the Amazon Redshift console, see Resizing a cluster.

The following example CLI command describes the configuration options available. In this example, the cluster named `mycluster` is a `rg.4xl` 8-node cluster.

```
aws redshift describe-node-configuration-options --cluster-identifier mycluster --
region eu-west-1 --action-type resize-cluster
```

This command returns an option list with recommended node types, number of nodes, and disk utilization for each option. The configurations returned can vary based on the specific input cluster. You can choose one of the returned configurations when you specify the options of the `resize-cluster` CLI command.

- *Ceiling on additional nodes* - Elastic resize has limits on the nodes that you can add to a cluster. For example, a RG or RA3 cluster supports elastic resize up to four times the number of nodes. To illustrate, you can add a node to a 4-node rg.4xl cluster to make it a five-node cluster, or add more nodes until it reaches sixteen.

  > **Note**
  >
  > The growth and reduction limits are based on the original node type and the number of nodes in the original cluster or its last classic resize. If an elastic resize will exceed the growth or reduction limits, use a classic resize.

  With some rg and ra3 node types, you can increase the number of nodes up to four times the existing count. Specifically, suppose that your cluster consists of ra3.4xlarge or ra3.16xlarge nodes. You can then use elastic resize to increase the number of nodes in an 8-node cluster to 32. Or you can pick a value below the limit. (Keep in mind that the ability to grow the cluster by 4x depends on the source cluster size.) If your cluster has ra3.xlplus nodes, the limit is double.

  All rg and ra3 node types support a decrease in the number of nodes to a quarter of the existing count. For example, you can decrease the size of a cluster with ra3.4xlarge nodes from 12 nodes to 3, or to a number above the minimum.

  The following table lists growth and reduction limits for each node type that supports elastic resize.

Original node type	Growth limit	Reduction limit
rg.12xlarge	4x	To one quarter of the number
rg.4xlarge	4x	To one quarter of the number
rg.xlarge	2x	To one half of the number
rg.large	2x	To one half of the number
ra3.16xlarge	4x (from 4 to 16 nodes, for example)	To one quarter of the number (from 16 to 4 nodes, for example)
ra3.4xlarge	4x	To one quarter of the number
ra3.xlplus	2x (from 4 to 8 nodes, for example)	To one quarter of the number
ra3.large	2x	To one half of the number
dc2.8xlarge	2x	To one half of the number (from 16 to 8 nodes, for example)
dc2.large	2x	To one half of the number

Note

Choosing legacy node types when you resize an RG or RA3 cluster – If you attempt to resize from a cluster with RG or RA3 nodes to another node type, such as DC2 , a validation warning message appears in the console, and the resize operation won't complete. This occurs because resize to legacy node types isn't supported. This is

to prevent a customer from resizing to a node type that's deprecated or soon to be deprecated. This applies for both elastic resize and classic resize.

Classic resize

Classic resize handles use cases where the change in cluster size or node type isn't supported by elastic resize. When you perform a classic resize, Amazon Redshift creates a target cluster and migrates your data and metadata to it from the source cluster.

Classic resize to RG or RA3 can provide better availability

Classic resize has been enhanced when the target node type is RG or RA3. It does this by using a backup and restore operation between the source and target cluster. When the resize begins, the source cluster restarts and is unavailable for a few minutes. After that, the cluster is available for read and write operations while the resize continues in the background.

Checking your cluster

To ensure you have the best performance and results when you perform a classic resize to an RG or RA3 cluster, complete this checklist. When you don't follow the checklist, you may not get some of the benefits of classic resizing with RG or RA3 nodes, such as the ability to do read and write operations.

1. The size of the data must be below 2 petabytes. (A petabyte is equal to 1,000 terabytes.)
   To validate the size of your data, create a snapshot and check its size. You can also run the following query to check the size:

   ```
   SELECT
   sum(case when lower(diststyle) like ('%key%') then size else 0 end) distkey_blocks,
   sum(size) as total_blocks,
   ((distkey_blocks/(total_blocks*1.00)))*100 as Blocks_need_redist
   FROM svv_table_info;
   ```

   The `svv_table_info` table is visible only to superusers.
2. Before you initiate a classic resize, make sure you have a manual snapshot that is no more than 10 hours old. If not, take a snapshot.
3. The snapshot used to perform the classic resize can't be used for a table restore or other purpose.

4. The cluster must be in a VPC.

Sorting and distribution operations that result from classic resize to RG or RA3

During classic resize to RG or RA3, tables with KEY distribution that are migrated as EVEN distribution are converted back to their original distribution style. The duration of this is dependent on the size of the data and how busy your cluster is. Query workloads are given higher priority to run over data migration. For more information, see Distribution styles. Both reads and writes to the database work during this migration process, but it can take longer for queries to complete. However, concurrency scaling can boost performance during this time by adding resources for query workloads. You can see the progress of data migration by viewing results from the SYS_RESTORE_STATE and SYS_RESTORE_LOG views. More information about monitoring follows.

After the cluster is fully resized, the following sort behavior occurs:

- If the resize results in the cluster having more slices, KEY distribution tables become partially unsorted, but EVEN tables remain sorted. Additionally, the information about how much data is sorted may not be up to date, directly following the resize. After key recovery, automatic vacuum sorts the table over time.
- If the resize results in the cluster having fewer slices, both KEY distribution and EVEN distribution tables become partially unsorted. Automatic vacuum sorts the table over time.

For more information about automatic table vacuum, see Vacuuming tables. For more information about slices in compute nodes, see Data warehouse system architecture.

Classic resize steps when the target cluster is RG or RA3

Classic resize consists of the following steps, when the target cluster type is RG or RA3 and you've met the prerequisites detailed in the previous section.

1. Migration initiates from the source cluster to the target cluster. When the new, target cluster is provisioned, Amazon Redshift sends an event notification that the resize has started. It restarts your existing cluster, which closes all connections. If your existing cluster is a datasharing producer cluster, connections with consumer clusters are also closed. The restart takes a few minutes.
2. After the restart, the database is available for reads and writes. Additionally, data sharing resumes, which takes an additional few minutes.

3. Data is migrated to the target cluster. When the target node type is RG or RA3, reads and writes are available during data migration.
4. When the resize process nears completion, Amazon Redshift updates the endpoint to the target cluster, and all connections to the source cluster are dropped. The target cluster becomes the producer for data sharing.
5. The resize completes. Amazon Redshift sends an event notification.

You can view the resize progress on the Amazon Redshift console. The time it takes to resize a cluster depends on the amount of data.

> **Note**
>
> **Choosing legacy node types when you resize an RG or RA3 cluster** – If you attempt to resize from a cluster with RG or RA3 nodes to another node type, such as DC2 , a validation warning message appears in the console, and the resize operation won't complete. This occurs because resize to legacy node types isn't supported. This is to prevent a customer from resizing to a node type that's deprecated or soon to be deprecated. This applies for both elastic resize and classic resize.

Monitoring a classic resize when the target cluster is RG or RA3

To monitor a classic resize of a provisioned cluster in progress, including KEY distribution, use SYS_RESTORE_STATE. It shows the percentage completed for the table being converted. You must be a super user to access the data.

Drop tables that you don't need when you perform a classic resize. When you do this, existing tables can be distributed more quickly.

Classic resize steps when the target cluster isn't RG or RA3

Classic resize consists of the following, when the target node type is anything other than RG or RA3, like DC2, for instance.

1. Migration initiates from the source cluster to the target cluster. When the new, target cluster is provisioned, Amazon Redshift sends an event notification that the resize has started. It restarts your existing cluster, which closes all connections. If your existing cluster is a datasharing producer cluster, connections with consumer clusters are also closed. The restart takes a few minutes.

Note that any database relation, such as a table or materialized view, created with BACKUP NO is not retained during the classic resize. For more information, see CREATE MATERIALIZED VIEW.

2. Following the restart, the database is available as read only. Data sharing resumes, which takes an additional few minutes.
3. Data is migrated to the target cluster. The database remains read only.
4. When the resize process nears completion, Amazon Redshift updates the endpoint to the target cluster, and all connections to the source cluster are dropped. The target cluster becomes the producer for data sharing.
5. The resize completes. Amazon Redshift sends an event notification.

You can view the resize progress on the Amazon Redshift console. The time it takes to resize a cluster depends on the amount of data.

> **Note**
>
> It can take days or possibly weeks to resize a cluster with a large amount of data when the target cluster isn't RG or RA3, or it doesn't meet the prerequisites for an RA3 target cluster detailed in the previous section.
> Also note that used storage capacity for the cluster can go up after a classic resize. This is normal system behavior when the cluster has additional data slices that result from the classic resize. This use of additional capacity can occur even when the number of nodes in the cluster stays the same.

Elastic resize vs classic resize

The following table compares behavior between the two resize types.

Behavior	Elastic resize	Classic resize	Comme
System data retention	Elastic resize retains system log data.	Classic resize doesn't retain system tables and data.	If you have audit logging enabled in

Behavior	Elastic resize	Classic resize	Comme
			your source cluster, you can continue to access the logs in Amazon S3 or in CloudWatch, following a resize. You can keep or delete these logs as your data policies specify.

Behavior	Elastic resize	Classic resize	Comme
Changing node types	Elastic resize, when the node type doesn't change: In-place resize, and most queries are held. Elastic resize, with a new node type selected: A new cluster is created. Queries are dropped as the resize process completes.	Classic Resize: A new cluster is created. Queries are dropped during the resize process.	

Behavior	Elastic resize	Classic resize	Comme
Session and query retention	Elastic resize retains sessions and queries when the node type is the same in the source cluster and target. If you choose a new node type, queries are dropped.	Classic resize doesn't retain sessions and queries. Queries are dropped.	When queries are dropped, you can expect some performance degradation. It's best to perform a resize operation during a period of light use.

Behavior	Elastic resize	Classic resize	Comme
Cancelling a resize operation	You can't cancel an elastic resize.	You can cancel a classic resize operation before it completes by choosing **Cancel resize** from the cluster details in the Amazon Redshift console.	The amount of time it takes to cancel a resize depends on the stage of the resize operation when you cancel. When you do this, the cluster isn't available until the cancel operation completes. If

Behavior	Elastic resize	Classic resize	Comme
			the resize operation is in the final stage, you can't cancel. For classic resize to an RG or RA3 cluster, you can't cancel.

Scheduling a resize

You can schedule resize operations for your cluster to scale up to anticipate high use or to scale down for cost savings. Scheduling works for both elastic resize and classic resize. You can set up a schedule on the Amazon Redshift console. For more information, see Resizing a cluster, under **Managing clusters using the console**. You can also use AWS CLI or Amazon Redshift API operations to schedule a resize. For more information, see create-scheduled-action in the *AWS CLI Command Reference* or CreateScheduledAction in the *Amazon Redshift API Reference*.

Snapshot, restore, and resize

Elastic resize is the fastest method to resize an Amazon Redshift cluster. If elastic resize isn't an option for you and you require near-constant write access to your cluster, use the snapshot and restore operations with classic resize as described in the following section. This approach requires

that any data that is written to the source cluster after the snapshot is taken must be copied manually to the target cluster after the switch. Depending on how long the copy takes, you might need to repeat this several times until you have the same data in both clusters. Then you can make the switch to the target cluster. This process might have a negative impact on existing queries until the full set of data is available in the target cluster. However, it minimizes the amount of time that you can't write to the database.

The snapshot, restore, and classic resize approach uses the following process:

1. Take a snapshot of your existing cluster. The existing cluster is the source cluster.
2. Note the time that the snapshot was taken. Doing this means that you can later identify the point when you need to rerun extract, transact, load (ETL) processes to load any post-snapshot data into the target database.
3. Restore the snapshot into a new cluster. This new cluster is the target cluster. Verify that the sample data exists in the target cluster.
4. Resize the target cluster. Choose the new node type, number of nodes, and other settings for the target cluster.
5. Review the loads from your ETL processes that occurred after you took a snapshot of the source cluster. Be sure to reload the same data in the same order into the target cluster. If you have ongoing data loads, repeat this process several times until the data is the same in both the source and target clusters.
6. Stop all queries running on the source cluster. To do this, you can reboot the cluster, or you can log on as a superuser and use the PG_CANCEL_BACKEND and the PG_TERMINATE_BACKEND commands. Rebooting the cluster is the easiest way to make sure that the cluster is unavailable.
7. Rename the source cluster. For example, rename it from `examplecluster` to `examplecluster-source`.
8. Rename the target cluster to use the name of the source cluster before the rename. For example, rename the target cluster from preceding to `examplecluster`. From this point on, any applications that use the endpoint containing `examplecluster` connect to the target cluster.
9. Delete the source cluster after you switch to the target cluster, and verify that all processes work as expected.

Alternatively, you can rename the source and target clusters before reloading data into the target cluster. This approach works if you don't require that any dependent systems and reports be

immediately up to date with those for the target cluster. In this case, step 6 moves to the end of the process described preceding.

The rename process is only required if you want applications to continue using the same endpoint to connect to the cluster. If you don't require this, you can instead update any applications that connect to the cluster to use the endpoint of the target cluster without renaming the cluster.

There are a couple of benefits to reusing a cluster name. First, you don't need to update application connection strings because the endpoint doesn't change, even though the underlying cluster changes. Second, related items such as Amazon CloudWatch alarms and Amazon Simple Notification Service (Amazon SNS) notifications are tied to the cluster name. This tie means that you can continue using the same alarms and notifications that you set up for the cluster. This continued use is primarily a concern in production environments where you want the flexibility to resize the cluster without reconfiguring related items, such as alarms and notifications.

Renaming a cluster

You can rename a cluster if you want the cluster to use a different name. Because the endpoint to your cluster includes the cluster name (also referred to as the *cluster identifier*), the endpoint changes to use the new name after the rename finishes. For example, if you have a cluster named `examplecluster` and rename it to `newcluster`, the endpoint changes to use the `newcluster` identifier. Any applications that connect to the cluster must be updated with the new endpoint.

You may rename a cluster if you want to change the cluster your applications connect to without having to change the endpoint in those applications. In this case, you must first rename the original cluster and then change the second cluster to reuse the name of the original cluster before the rename. Doing this is necessary because the cluster identifier must be unique within your account and region, so the original cluster and second cluster cannot have the same name. You might do this if you restore a cluster from a snapshot and don't want to change the connection properties of any dependent applications.

> **Note**
>
> If you delete the original cluster, you are responsible for deleting any unwanted cluster snapshots.

When you rename a cluster, the cluster status changes to `renaming` until the process finishes. The old DNS name that was used by the cluster is immediately deleted, although it could remain

cached for a few minutes. The new DNS name for the renamed cluster becomes effective within about 10 minutes. The renamed cluster is not available until the new name becomes effective. The cluster will be rebooted and any existing connections to the cluster will be dropped. After this completes, the endpoint will change to use the new name. For this reason, you should stop queries from running before you start the rename and restart them after the rename finishes.

Cluster snapshots are retained, and all snapshots associated with a cluster remain associated with that cluster after it is renamed. For example, suppose that you have a cluster that serves your production database and the cluster has several snapshots. If you rename the cluster and then replace it in the production environment with a snapshot, the cluster that you renamed still has those existing snapshots associated with it.

Amazon CloudWatch alarms and Amazon Simple Notification Service (Amazon SNS) event notifications are associated with the name of the cluster. If you rename the cluster, you must update these accordingly. You can update the CloudWatch alarms in the CloudWatch console, and you can update the Amazon SNS event notifications in the Amazon Redshift console on the **Events** pane. The load and query data for the cluster continues to display data from before the rename and after the rename. However, performance data is reset after the rename process finishes.

For more information, see Modifying a cluster.

Upgrading the release version of a cluster

You can upgrade the release maintenance version of a cluster that has a **Release Status** value of **New release available**. When you upgrade the maintenance version, you can choose to upgrade immediately or upgrade in the next maintenance window.

> **Important**
>
> If you upgrade immediately, your cluster is offline until the upgrade completes.

To upgrade a cluster to a new release version

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**.
3. Choose the cluster to upgrade.

4. For **Actions**, choose **Upgrade cluster version**. The **Upgrade cluster version** page appears.
5. Follow the instructions on the page.
6. Choose **Upgrade cluster version**.

Pausing and resuming a cluster

If you have a cluster that only needs to be available at specific times, you can pause the cluster and later resume it. While the cluster is paused, on-demand billing is suspended. Only the cluster's storage incurs charges. For more information about pricing, see the Amazon Redshift pricing page.

When you pause a cluster, Amazon Redshift creates a snapshot, begins terminating queries, and puts the cluster in a pausing state. If you delete a paused cluster without requesting a final snapshot, then you can't restore the cluster. You can't cancel or roll back a pause or resume operation after it's initiated.

You can pause and resume a cluster on the Amazon Redshift console, with the AWS CLI, or with Amazon Redshift API operations.

You can schedule actions to pause and resume a cluster. When you use the new Amazon Redshift console to create a recurring schedule to pause and resume, then two scheduled actions are created for the date range that you choose. The scheduled action names are suffixed with `-pause` and `-resume`. The total length of the name must fit within the maximum size of a scheduled action name.

You can't pause the following types of clusters:

- EC2-Classic clusters.
- Clusters that are not active, for example, a cluster that is currently modifying.
- Hardware security module (HSM) clusters.
- Clusters that have automated snapshots turned off.

When deciding to pause a cluster, consider the following:

- Connections or queries to the cluster aren't available.
- You can't see query monitoring information of a paused cluster on the Amazon Redshift console.
- You can't modify a paused cluster. Any scheduled actions on the cluster aren't done. These include creating snapshots, resizing clusters, and cluster maintenance operations.

- Hardware metrics aren't created. Update your CloudWatch alarms if you have alarms set on missing metrics.
- You can't copy the latest automated snapshots of a paused cluster to manual snapshots.
- While a cluster is pausing, it can't be resumed until the pause operation is complete.
- When you pause a cluster, billing is suspended. However, the pause operation typically completes within 15 minutes, depending upon the size of the cluster.
- Audit logs are archived and not restored on resume.
- After a cluster is paused, traces and logs might not be available for troubleshooting problems that occurred before the pause.
- If you're managing your admin credentials using AWS Secrets Manager and pause your cluster, your cluster's secret won't be deleted and you'll continue to be billed for the secret. For more information on managing your Redshift admin password with AWS Secrets Manager, see Managing Amazon Redshift admin passwords using AWS Secrets Manager.
- No-backup tables on the cluster are restored on resume for RG and RA3 instance types. They aren't restored on resume for DC2 instance types. For more information about no-backup tables, see Excluding tables from snapshots.

When you resume a cluster, consider the following:

- The cluster version of the resumed cluster is updated to the maintenance version based on the maintenance window of the cluster.
- If you delete the subnet associated with a paused cluster, you might have an incompatible network. In this case, restore your cluster from the latest snapshot.
- If you delete an Elastic IP address while the cluster is paused, then a new Elastic IP address is requested.
- If Amazon Redshift can't resume the cluster with its previous elastic network interface, then Amazon Redshift tries to allocate a new one.
- When you resume a cluster, your node IP addresses might change. You might need to update your VPC settings to support these new IP addresses for features like COPY from Secure Shell (SSH) or COPY from Amazon EMR.
- If you try to resume a cluster that isn't paused, the resume operation returns an error. If the resume operation is part of a scheduled action, modify or delete the scheduled action to prevent future errors.

- Depending upon the size of the cluster, it can take several minutes to resume a cluster before queries can be processed. In addition, query performance can be impacted for some period of time while the cluster is being re-hydrated after resume completes.

Rebooting a cluster

Rebooting a cluster is a cluster operation that restarts the cluster with the same configuration as before the reboot. You can reboot a cluster to apply pending maintenance updates, reset configuration changes, recover from certain issues, or troubleshoot cluster problems. Rebooting a cluster can help ensure optimal performance, security, and stability of the Amazon Redshift environment. The following procedure provides detailed steps for rebooting an Amazon Redshift cluster.

When you reboot a cluster, the cluster status is set to `rebooting` and a cluster event is created when the reboot is completed. Any pending cluster modifications are applied at this reboot.

To reboot a cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**.
3. Choose the cluster to reboot.
4. For **Actions**, choose **Reboot cluster**. The **Reboot cluster** page appears.
5. Choose **Reboot cluster**.

Relocating a cluster

By using *relocation* in Amazon Redshift, you allow Amazon Redshift to move a cluster to another Availability Zone (AZ) without any loss of data or changes to your applications. With relocation, you can continue operations when there is an interruption of service on your cluster with minimal impact.

When cluster relocation is turned on, Amazon Redshift might choose to relocate clusters in some situations. In particular, this happens where issues in the current Availability Zone prevent optimal cluster operation or to improve service availability. You can also invoke the relocation function in cases where resource constraints in a given Availability Zone are disrupting cluster operations. An

example is the ability to resume or resize a cluster. Amazon Redshift offers the relocation feature at no extra charge.

When an Amazon Redshift cluster is relocated to a new Availability Zone, the new cluster has the same endpoint as the original cluster. Your applications can reconnect to the endpoint and continue operations without modifications or loss of data. However, relocation might not always be possible due to potential resource constraints in a given Availability Zone.

Amazon Redshift cluster relocation is supported for the RG and RA3 instance types. RG and RA3 instance types use Redshift Managed Storage (RMS) as a durable storage layer. The latest copy of a cluster's data is always available in other Availability Zones in an AWS Region. In other words, you can relocate an Amazon Redshift cluster to another Availability Zone without any loss of data.

When you turn on relocation for your cluster, Amazon Redshift migrates your cluster to be behind a proxy. Doing this helps implement location-independent access to a cluster's compute resources. The migration causes the cluster to be rebooted. When a cluster is relocated to another Availability Zone, an outage occurs while the new cluster is brought back online in the new Availability Zone. However, you don't have to make any changes to your applications because the cluster endpoint remains unchanged even after the cluster is relocated to the new Availability Zone.

Cluster relocation is enabled by default on newly created or restored RG or RA3 clusters whose subnet group includes multiple Availability Zones. Amazon Redshift assigns 5439 as the default port while creating a provisioned cluster. You can change to another port from the port range of 5431-5455 or 8191-8215. (Don't change to a port outside the ranges. It results in an error.) To change the default port for a provisioned cluster, use the Amazon Redshift console, AWS CLI, or Amazon Redshift API. To change the default port for a serverless workgroup, use the AWS CLI or the Amazon Redshift Serverless API.

If you turn on relocation and you currently use the leader node IP address to access your cluster or Enhanced VPC Routing, make sure to change that access. Instead, use the IP address associated with the cluster's virtual private cloud (VPC) endpoint. To find this cluster IP address, find and use the VPC endpoint in the **Network and security** section of the cluster details page. To get more details on the VPC endpoint, sign in to the Amazon VPC console.

You can also use the AWS Command Line Interface (AWS CLI) command `describe-vpc-endpoints` to get the elastic network interface associated with the endpoint. You can use the `describe-network-interfaces` command to get the associated IP address. For more information on Amazon Redshift AWS CLI commands, see Available commands in the *AWS CLI Command Reference.*

Limitations

When using Amazon Redshift relocation, be aware of the following limitations:

- Cluster relocation might not be possible in all scenarios due to potential resource limitations in a given Availability Zone. If this happens, Amazon Redshift doesn't change the original cluster.
- Relocation isn't supported on DC2 instance families of products.
- You can't perform a relocation across AWS Regions.
- Amazon Redshift relocation defaults to port number 5439. You can also change to another port in the ranges 5431-5455 or 8191-8215.

Managing relocation using the console

You can manage the settings for cluster relocation using the Amazon Redshift console.

Turning off relocation when creating a new cluster

Use the following procedure to turn off relocation when creating a new cluster.

To turn off relocation for a new cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**.
3. Choose **Create cluster** to create a new cluster. For more information on how to create a cluster, see Get started with Amazon Redshift provisioned data warehouses in *Amazon Redshift Getting Started Guide*.
4. Under **Backup**, for **Cluster relocation**, choose **Disabled**. Relocation is turned on by default.
5. Choose **Create cluster**.

Modifying relocation for an existing cluster

Use the following procedure to change the relocation setting for an existing cluster.

To modify the relocation setting for an existing cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.

2. On the navigation menu, choose **Clusters**. The clusters for your account in the current AWS Region are listed. A subset of properties of each cluster is displayed in columns in the list.
3. Choose the name of the cluster that you want to modify from the list. The cluster details page appears.
4. Choose the **Maintenance** tab, then in the **Backup details** section choose **Edit**.
5. Under **Backup**, choose **Disabled**. Relocation is turned on by default.
6. Choose **Modify cluster**.

Relocating a cluster

Use the following procedure to manually relocate a cluster to another Availability Zone. This is especially useful when you want to test your network setup in secondary Availability Zones or when you are running into resource constraints in the current Availability Zone.

To relocate a cluster to another Availability Zone

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**. The clusters for your account in the current AWS Region are listed. A subset of properties of each cluster is displayed in columns in the list.
3. Choose the name of the cluster that you want to move from the list. The cluster details page appears.
4. For **Actions**, choose **Relocate**. The **Relocate cluster** page appears.
5. (Optional) Choose an **Availability Zone**. If you don't choose an Availability Zone, Amazon Redshift chooses one for you.

Amazon Redshift starts the relocation and displays the cluster as relocating. After the relocation completes, the cluster status changes to available.

Managing relocation using the Amazon Redshift CLI

You can manage the settings for cluster relocation using the AWS Command Line Interface (CLI).

With the AWS CLI, the following example command creates an Amazon Redshift cluster named **`mycluster`** that has relocation turned on.

```
aws redshift create-cluster --cluster-identifier mycluster --number-of-nodes 2 --
master-username enter a username --master-user-password enter a password --node-type
 ra3.4xlarge --port 5439 --no-availability-zone-relocation
```

If your current cluster is using a different port, you must modify it to use from the port range of 5431-5455 or 8191-8215 before modifying it to turn on relocation. The default is 5439. The following example command modifies the port in case your cluster doesn't use one from the given range.

```
aws redshift modify-cluster --cluster-identifier mycluster --port 5439
```

The following example command includes the availability-zone-relocation parameter on the Amazon Redshift cluster.

```
aws redshift modify-cluster --cluster-identifier mycluster --availability-zone-
relocation
```

The following example command turns off the availability-zone-relocation parameter on the Amazon Redshift cluster.

```
aws redshift modify-cluster --cluster-identifier mycluster --no-availability-zone-
relocation
```

The following example command invokes relocation on the Amazon Redshift cluster.

```
aws redshift modify-cluster --cluster-identifier mycluster --availability-zone us-
east-1b
```

Setting a usage limit on a cluster

You can add up to four usage limits to control usage for each of the following:

- Concurrency scaling
- Automatic optimizations run using extra compute resources
- Redshift Spectrum usage
- Cross-Region data sharing

Setting a usage limit for a provisioned cluster

Following is the procedure for setting a usage limit on a provisioned cluster:

To set a usage limit for a cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. Navigate to the provisioned cluster that you want to set a limit for.
3. From the cluster's details page, select **Manage usage limit** from the **Actions** drop-down menu. You can also select the **Maintenance** tab for a cluster, then scroll down and select **Create usage limits**.
4. Select **Add limit** for the usage limit that you want to set. You can add up to 4 limits for a given feature.
5. Set a **Time period** for the usage limit, which is either **Daily**, **Weekly**, or **Monthly**.
6. Set a **Usage limit**.

   - For concurrency scaling and automatic optimizations run using extra compute resources limits, the usage limit is the amount of time that Amazon Redshift spends using the feature in the given time period. In this case, the usage limit is set in hours and minutes.
   - For Redshift Spectrum, the usage limit is the amount of data scanned from Amazon S3. In this case, the usage limit is set in terabytes (TB).
   - For cross-Region data sharing, the usage limit is the amount of data transferred from the producer Region to consumer Regions that consumers can query. In this case, the usage limit is set in terabytes (TB).
7. Set the **Action** for Amazon Redshift to take when your cluster reaches the limit. These are the following:

   - **Log to system table** - Adds a record to the system view SYS_QUERY_HISTORY. You can query the usage_limit column in this view to determine if a query exceeded the limit.
   - **Alert** - Uses Amazon SNS to set up notification subscriptions and send notifications if a limit is breached. You can choose an existing Amazon SNS topic, create a new topic, or proceed without one.
   - **Disable feature** - Disables the feature. You can also choose to use Amazon SNS to send a notification. Users can continue to use the cluster for other tasks.

The first two actions are informational, but the last turns off use of the feature.

8. Choose **Save changes** at the bottom of the page to save the limit. If you set more than one limit at once, **Save changes** will save all of them at once.

Shutting down and deleting a cluster

You can shut down your cluster if you want to stop it from running and incurring charges. When you shut it down, you can optionally create a final snapshot. If you create a final snapshot, Amazon Redshift will create a manual snapshot of your cluster before shutting it down. If you plan to provision a new cluster with the same data and configuration as the one you are deleting, you need a manual snapshot. By using a manual snapshot, you can restore the snapshot later and resume using the cluster.

If you no longer need your cluster and its data, you can shut it down without creating a final snapshot. In this case, the cluster and data are deleted permanently.

Regardless of whether you shut down your cluster with a final manual snapshot, all automated snapshots associated with the cluster will be deleted after the cluster is shut down. Any manual snapshots associated with the cluster are retained. Any manual snapshots that are retained, including the optional final snapshot, are charged at the Amazon Simple Storage Service storage rate if you have no other clusters running when you shut down the cluster, or if you exceed the available free storage that is provided for your running Amazon Redshift clusters. For more information about snapshot storage charges, see the Amazon Redshift pricing page.

Deleting a cluster also deletes any associated AWS Secrets Manager secrets.

To delete a cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**.
3. Choose the cluster to delete.
4. For **Actions**, choose **Delete**. The **Delete cluster** page appears.
5. Choose **Delete cluster**.

> **Note**
>
> When you delete a cluster and choose to create a final snapshot, Amazon Redshift will stop the delete request if a restore operation is in progress on the cluster. If this occurs, you can delete the cluster without a final snapshot, or you can delete it with a final snapshot after the restore completes.

Amazon Redshift snapshots and backups

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 S3 by using an encrypted Secure Sockets Layer (SSL) connection.

Amazon Redshift automatically takes incremental snapshots that track changes to the cluster since the previous automated snapshot. Automated snapshots retain all of the data required to restore a cluster from a snapshot. You can create a snapshot schedule to control when automated snapshots are taken, or you can take a manual snapshot any time.

When you restore from a snapshot, Amazon Redshift creates a new cluster and makes the new cluster available before all of the data is loaded, so you can begin querying the new cluster immediately. The cluster streams data on demand from the snapshot in response to active queries, then loads the remaining data in the background.

When you launch a cluster, you can set the retention period for automated and manual snapshots. You can change the default retention period for automated and manual snapshots by modifying the cluster. You can change the retention period for a manual snapshot when you create the snapshot or by modifying the snapshot.

You can monitor the progress of snapshots by viewing the snapshot details in the AWS Management Console, or by calling describe-cluster-snapshots in the CLI or the DescribeClusterSnapshots API action. For an in-progress snapshot, these display information such as the size of the incremental snapshot, the transfer rate, the elapsed time, and the estimated time remaining.

To ensure that your backups are always available to your cluster, Amazon Redshift stores snapshots in an internally managed Amazon S3 bucket that is managed by Amazon Redshift. To manage storage charges, evaluate how many days you need to keep automated snapshots and configure

their retention period accordingly. Delete any manual snapshots that you no longer need. For more information about the cost of backup storage, see the Amazon Redshift pricing page.

You can also create and restore snapshots using AWS Backup, a fully managed service that helps you centralize and automate data protection across AWS services, in the cloud, and on premises. For more information, see AWS Backup integration with Amazon Redshift. For information on AWS Backup, see What is AWS Backup? in the *AWS Backup Developer Guide*.

Working with snapshots and backups in Amazon Redshift Serverless

Amazon Redshift Serverless, like a provisioned cluster, enables you to take a backup as a point-in-time representation of the objects and data in the namespace. There are two types of backups in Amazon Redshift Serverless: snapshots that are manually created and recovery points that Amazon Redshift Serverless creates automatically. You can find more information about working with snapshots for Amazon Redshift Serverless at Snapshots and recovery points.

You can also restore a snapshot from a provisioned cluster to a serverless namespace. For more information, see Restoring a serverless namespace from a snapshot.

Automated snapshots

When automated snapshots are enabled for a cluster, Amazon Redshift periodically takes snapshots of that cluster. By default Amazon Redshift takes a snapshot about every eight hours or following every 5 GB per node of data changes, or whichever comes first. If your data is larger than 5 GB * the number of nodes, the shortest amount of time in between automated snapshot creation is 15 minutes. Alternatively, you can create a snapshot schedule to control when automated snapshots are taken. If you're using custom schedules, the minimum amount of time between automated snapshots is one hour. Automated snapshots are enabled by default when you create a cluster.

Note

Amazon Redshift takes automated snapshots of a paused data sharing producer cluster. Because consumers can write to a paused producer, these snapshots capture those data sharing writes and provide a recovery point. Previously, Amazon Redshift didn't take automated snapshots of these clusters while they were paused. These snapshots follow the cluster's existing automated snapshot schedule and retention period, and no action is required. For a paused cluster that isn't a data sharing producer, Amazon Redshift doesn't take automated snapshots while the cluster is paused.

Automated snapshots are deleted at the end of a retention period. The default retention period is one day, but you can modify it by using the Amazon Redshift console or programmatically by using the Amazon Redshift API or CLI.

To disable automated snapshots, set the retention period to zero. If you disable automated snapshots, Amazon Redshift stops taking snapshots and deletes any existing automated snapshots for the cluster. You can't disable automated snapshots for RG or RA3 node types. You can set an RG or RA3 node type automated retention period from 1–35 days.

Only Amazon Redshift can delete an automated snapshot; you cannot delete them manually. Amazon Redshift deletes automated snapshots at the end of a snapshot's retention period, when you disable automated snapshots for the cluster, or when you delete the cluster. Amazon Redshift retains the latest automated snapshot until you disable automated snapshots or delete the cluster.

If you want to keep an automated snapshot for a longer period, you can create a copy of it as a manual snapshot. The automated snapshot is retained until the end of the retention period, but the corresponding manual snapshot is retained until you manually delete it or until the end of the retention period.

Automated snapshot schedules

To precisely control when snapshots are taken, you can create a snapshot schedule and attach it to one or more clusters. When you modify a snapshot schedule, the schedule is modified for all associated clusters. If a cluster doesn't have a snapshot schedule attached, the cluster uses the default automated snapshot schedule.

A *snapshot schedule* is a set of schedule rules. You can define a simple schedule rule based on a specified interval, such as every 8 hours or every 12 hours. You can also add rules to take snapshots on certain days of the week, at specific times, or during specific periods. Rules can also be defined using Unix-like cron expressions.

Snapshot schedule format

On the Amazon Redshift console, you can create a snapshot schedule. Then, you can attach a schedule to a cluster to trigger the creation of a system snapshot. A schedule can be attached to multiple clusters, and you can create multiple cron definitions in a schedule to trigger a snapshot.

You can define a schedule for your snapshots using a cron syntax. The definition of these schedules uses a modified Unix-like cron syntax. You specify time in Coordinated universal time (UTC). You

can create schedules with a maximum frequency of one hour and minimum precision of one minute.

Amazon Redshift modified cron expressions have 3 required fields, which are separated by white space.

Syntax

```
cron(Minutes Hours Day-of-month Month Day-of-week Year)
```

Fields	Values	Wildcards
Minutes	0–59	, - * /
Hours	0–23	, - * /
Day-of-month	1–31	, - * ? / L W
Month	1–12 or JAN-DEC	, - * /
Day-of-week	1–7 or SUN-SAT	, - * ? L #
Year	1970–2199	, - * /

Wildcards

- The **,** (comma) wildcard includes additional values. In the `Day-of-week` field, `MON,WED,FRI` would include Monday, Wednesday, and Friday. Total values are limited to 24 per field.
- The **-** (dash) wildcard specifies ranges. In the `Hour` field, 1–15 would include hours 1 through 15 of the specified day.
- The ***** (asterisk) wildcard includes all values in the field. In the `Hours` field, ***** would include every hour.
- The **/** (forward slash) wildcard specifies increments. In the `Hours` field, you could enter **1/10** to specify every 10th hour, starting from the first hour of the day (for example, the 01:00, 11:00, and 21:00).
- The **?** (question mark) wildcard specifies one or another. In the `Day-of-month` field you could enter **7**, and if you didn't care what day of the week the seventh was, you could enter **?** in the Day-of-week field.

- The **L** wildcard in the `Day-of-month` or `Day-of-week` fields specifies the last day of the month or week.
- The **W** wildcard in the `Day-of-month` field specifies a weekday. In the `Day-of-month` field, `3W` specifies the day closest to the third weekday of the month.
- The **#** wildcard in the Day-of-week field specifies a certain instance of the specified day of the week within a month. For example, 3#2 would be the second Tuesday of the month: the 3 refers to Tuesday because it is the third day of each week, and the 2 refers to the second day of that type within the month.

> **Note**
>
> If you use a '#' character, you can define only one expression in the day-of-week field. For example, "3#1,6#3" is not valid because it is interpreted as two expressions.

Limits

- You can't specify the `Day-of-month` and `Day-of-week` fields in the same cron expression. If you specify a value in one of the fields, you must use a **?** (question mark) in the other.
- Snapshot schedules don't support the following frequencies:
  - Snapshots scheduled more frequently than 1 per hour.
  - Snapshots scheduled less frequently than 1 per day (24 hours).

  If you have overlapping schedules that result in scheduling snapshots within a 1 hour window, a validation error results.

When creating a schedule, you can use the following sample cron strings.

Minutes	Hours	Day of week	Meaning			
0	14-20/1	TUE	Every hour between 2pm and 8pm on Tuesday.			

Minutes	Hours	Day of week	Meaning
0	21	MON-FRI	Every night at 9pm Monday–Friday.
30	0/6	SAT-SUN	Every 6 hour increment on Saturday and Sunday starting at 30 minutes after midnight (00:30) that day. This results in a snapshot at [00:30, 06:30, 12:30, and 18:30] each day.
30	12/4	*	Every 4 hour increment starting at 12:30 each day. This resolves to [12:30, 16:30, 20:30].

For example to run on a schedule on an every 2 hour increment starting at 15:15 each day. This resolves to [15:15, 17:15, 19:15, 21:15, 23:15] , specify:

```
cron(15 15/2 *)
```

You can create multiple cron schedule definitions within as schedule. For example the following AWS CLI command contains two cron schedules in one schedule.

```
create-snapshot-schedule --schedule-identifier "my-test" --schedule-definition "cron(0
 17 SAT,SUN)" "cron(0 9,17 MON-FRI)"
```

Manual snapshots

You can take a manual snapshot any time. By default, manual snapshots are retained indefinitely, even after you delete your cluster. You can specify the retention period when you create a manual snapshot, or you can change the retention period by modifying the snapshot. For more information about changing the retention period, see Modifying the manual snapshot retention period.

If a snapshot is deleted, you can't start any new operations that reference that snapshot. However, if a restore operation is in progress, that restore operation will run to completion.

Amazon Redshift has a quota that limits the total number of manual snapshots that you can create; this quota is per AWS account per AWS Region. The default quota is listed at Quotas and limits in Amazon Redshift.

Snapshot storage

Because snapshots accrue storage charges, it's important that you delete them when you no longer need them. Amazon Redshift deletes automated and manual snapshots at the end of their respective snapshot retention periods. You can also delete manual snapshots using the AWS Management Console or with the batch-delete-cluster-snapshots CLI command.

You can change the retention period for a manual snapshot by modifying the manual snapshot settings.

You can get information about how much storage your snapshots are consuming using the Amazon Redshift Console or using the describe-storage CLI command.

Excluding tables from snapshots

By default, all user-defined permanent tables are included in snapshots. If a table, such as a staging table, doesn't need to be backed up, you can significantly reduce the time needed to create snapshots and restore from snapshots. You also reduce storage space on Amazon S3 by using a no-backup table. To create a no-backup table, include the BACKUP NO parameter when you create the table. For more information, see CREATE TABLE and CREATE TABLE AS in the *Amazon Redshift Database Developer Guide*.

> **Note**
>
> No-backup tables aren't supported for RG or RA3 provisioned clusters and Amazon Redshift Serverless workgroups. A table marked as no-backup in an RG or RA3 cluster or serverless

workgroup is treated as a permanent table that will always be backed up while taking a snapshot, and always restored when restoring from a snapshot. To avoid snapshot costs for no-backup tables, truncate them before taking a snapshot.

Creating a manual snapshot

You can create a manual snapshot of a cluster from the snapshots list as follows. Or, you can take a snapshot of a cluster in the cluster configuration pane. For more information, see Amazon Redshift snapshots and backups.

Note

No-backup tables aren't supported for RG or RA3 provisioned clusters and Amazon Redshift Serverless workgroups. A table marked as no-backup in an RG or RA3 cluster or serverless workgroup is treated as a permanent table that will always be backed up while taking a snapshot, and always restored when restoring from a snapshot. To avoid snapshot costs for no-backup tables, truncate them before taking a snapshot.

To create a manual snapshot

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, **Snapshots**, then choose **Create snapshot**. The snapshot page to create a manual snapshot is displayed.
3. Enter the properties of the snapshot definition, then choose **Create snapshot**. It might take some time for the snapshot to be available.

Creating a snapshot schedule

Amazon Redshift takes automatic, incremental snapshots of your data periodically and saves them to Amazon S3. Additionally, you can take manual snapshots of your data whenever you want.

All snapshot tasks in the Amazon Redshift console start from the snapshot list. You can filter the list by using a time range, the snapshot type, and the cluster associated with the snapshot. In addition, you can sort the list by date, size, and snapshot type. Depending on the snapshot type that you select, you might have different options available for working with the snapshot.

To precisely control when snapshots are taken, you can create a snapshot schedule and attach it to one or more clusters. You can attach a schedule when you create a cluster or by modifying the cluster. For more information, see Automated snapshot schedules.

To create a snapshot schedule

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, **Snapshots**, then choose the **Snapshot schedules** tab. The snapshot schedules are displayed.
3. Choose **Add schedule** to display the page to add a schedule.
4. Enter the properties of the schedule definition, then choose **Add schedule**.
5. On the page that appears, you can attach clusters to your new snapshot schedule, then choose **OK**.

Sharing a snapshot

You can share an existing manual snapshot with other AWS customer accounts by authorizing access to the snapshot. You can authorize up to 20 for each snapshot and 100 for each AWS Key Management Service (AWS KMS) key. That is, if you have 10 snapshots that are encrypted with a single KMS key, then you can authorize 10 AWS accounts to restore each snapshot, or other combinations that add up to 100 accounts and do not exceed 20 accounts for each snapshot. A person logged in as a user in one of the authorized accounts can then describe the snapshot or restore it to create a new Amazon Redshift cluster under their account. For example, if you use separate AWS customer accounts for production and test, a user can log on using the production account and share a snapshot with users in the test account. Someone logged on as a test account user can then restore the snapshot to create a new cluster that is owned by the test account for testing or diagnostic work.

A manual snapshot is permanently owned by the AWS customer account under which it was created. Only users in the account owning the snapshot can authorize other accounts to access the snapshot, or to revoke authorizations. Users in the authorized accounts can only describe or restore any snapshot that has been shared with them; they cannot copy or delete snapshots that have been shared with them. An authorization remains in effect until the snapshot owner revokes it. If an authorization is revoked, the previously authorized user loses visibility of the snapshot and cannot launch any new actions referencing the snapshot. If the account is in the process of

restoring the snapshot when access is revoked, the restore runs to completion. You cannot delete a snapshot while it has active authorizations; you must first revoke all of the authorizations.

AWS customer accounts are always authorized to access snapshots owned by the account. Attempts to authorize or revoke access to the owner account will receive an error. You cannot restore or describe a snapshot that is owned by an inactive AWS customer account.

After you have authorized access to an AWS customer account, no users in that account can perform any actions on the snapshot unless they assume a role with policies that allow them to do so.

- Users in the snapshot owner account can authorize and revoke access to a snapshot only if they assume a role with an IAM policy that allows them to perform those actions with a resource specification that includes the snapshot. For example, the following policy allows a user or role in AWS account `012345678912` to authorize other accounts to access a snapshot named `my-snapshot20130829`:

  JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
          "redshift:AuthorizeSnapshotAccess",
          "redshift:RevokeSnapshotAccess"
          ],
      "Resource":[
           "arn:aws:redshift:us-east-1:012345678912:snapshot:*/my-
snapshot20130829"
          ]
    }
  ]
}
```

- Users in an AWS account with which a snapshot has been shared cannot perform actions on that snapshot unless they have permissions allowing those actions. You can do this by assigning the policy to a role and assuming the role.
  - To list or describe a snapshot, they must have an IAM policy that allows the `DescribeClusterSnapshots` action. The following code shows an example:

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
          "redshift:DescribeClusterSnapshots"
          ],
      "Resource":[
           "*"
          ]
    }
  ]
}
```

- To restore a snapshot, a user must assume a role with an IAM policy that allows the `RestoreFromClusterSnapshot` action and has a resource element that covers both the cluster they are attempting to create and the snapshot. For example, if a user in account `012345678912` has shared snapshot `my-snapshot20130829` with account `219876543210`, in order to create a cluster by restoring the snapshot, a user in account `219876543210` must assume a role with a policy such as the following:

  JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
          "redshift:RestoreFromClusterSnapshot"
          ],
      "Resource":[
           "arn:aws:redshift:us-east-1:012345678912:snapshot:*/my-
snapshot20130829",
           "arn:aws:redshift:us-east-1:219876543210:cluster:from-another-
account"
          ]
    }
  ]
```

```
}
```

- After access to a snapshot has been revoked from an AWS account, no users in that account can access the snapshot. This is the case even if those accounts have IAM policies that allow actions on the previously shared snapshot resource.

Sharing a cluster snapshot using the console

On the console, you can authorize other users to access a manual snapshot you own, and you can later revoke that access when it is no longer required.

To share a snapshot with another account

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, **Snapshots**, then choose the manual snapshot to share.
3. For **Actions**, choose **Manual snapshot settings** to display the properties of the manual snapshot.
4. Enter the account or accounts to share with in the **Manage access** section, then choose **Save**.

Security considerations for sharing encrypted snapshots

When you provide access to an encrypted snapshot, Redshift requires that the AWS KMS customer managed key used to create the snapshot is shared with the account or accounts performing the restore. If the key isn't shared, attempting to restore the snapshot results in an access-denied error. The receiving account doesn't need any extra permissions to restore a shared snapshot. When you authorize snapshot access and share the key, the identity authorizing access must have `kms:DescribeKey` permissions on the key used to encrypt the snapshot. This permission is described in more detail in AWS KMS permissions. For more information, see DescribeKey in the Amazon Redshift API reference documentation.

The customer managed key policy can be updated programmatically or in the AWS Key Management Service console.

> **Note**
>
> If you're using a default KMS key, you don't need to take action or change anything in AWS KMS in order to share a snapshot.

Allowing access to the AWS KMS key for an encrypted snapshot

To share the AWS KMS customer managed key for an encrypted snapshot, update the key policy by performing the following steps:

1. Update the KMS key policy with the Amazon Resource Name (ARN) of the AWS account that you are sharing to as `Principal` in the KMS key policy.
2. Allow the `kms:Decrypt` action.

In the following key-policy example, user `111122223333` is the owner of the KMS key, and user `444455556666` is the account that the key is shared with. This key policy gives the AWS account access to the sample KMS key by including the ARN for the root AWS account identity for user `444455556666` as a `Principal` for the policy, and by allowing the `kms:Decrypt` action.

JSON

```
{
    "Id": "key-policy-1",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::444455556666:root"
                ]
            },
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

After access is granted to the customer managed KMS key, the account that restores the encrypted snapshot must create an AWS Identity and Access Management (IAM) role, or user, if it doesn't already have one. In addition, that AWS account must also attach an IAM policy to that IAM role or user that allows them to restore an encrypted database snapshot, using your KMS key.

For more information about giving access to an AWS KMS key, see Allowing users in other accounts to use a KMS key, in the AWS Key Management Service developer guide.

For an overview of key policies, see How Amazon Redshift uses AWS KMS.

Copying an automated snapshot

Automated snapshots are automatically deleted when their retention period expires, when you disable automated snapshots, or when you delete a cluster. If you want to keep an automated snapshot, you can copy it to a manual snapshot.

To copy an automated snapshot

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, **Snapshots**, then choose the snapshot to copy.
3. For **Actions**, choose **Copy automated snapshot** to copy the snapshot.
4. Update the properties of the new snapshot, then choose **Copy**.

Copying a snapshot to another AWS Region

You can configure Amazon Redshift to automatically copy snapshots (automated or manual) for a cluster to another AWS Region. When a snapshot is created in the cluster's primary AWS Region, it's copied to a secondary AWS Region. The two AWS Regions are known respectively as the *source AWS Region* and *destination AWS Region*. If you store a copy of your snapshots in another AWS Region, you can restore your cluster from recent data if anything affects the primary AWS Region. You can configure your cluster to copy snapshots to only one destination AWS Region at a time. For a list of Amazon Redshift Regions, see Regions and endpoints in the *Amazon Web Services General Reference*.

When you enable Amazon Redshift to automatically copy snapshots to another AWS Region, you specify the destination AWS Region to copy the snapshots to. For automated snapshots, you can also specify the retention period to keep them in the destination AWS Region. After an automated snapshot is copied to the destination AWS Region and it reaches the retention time period there, it's deleted from the destination AWS Region. Doing this keeps your snapshot usage low. To keep the automated snapshots for a shorter or longer time in the destination AWS Region, change this retention period.

The retention period that you set for automated snapshots that are copied to the destination AWS Region is separate from the retention period for automated snapshots in the source AWS Region. The default retention period for copied snapshots is seven days. That seven-day period applies only to automated snapshots. In both the source and destination AWS Regions, manual snapshots are deleted at the end of the snapshot retention period or when you manually delete them.

You can disable automatic snapshot copy for a cluster at any time. When you disable this feature, snapshots are no longer copied from the source AWS Region to the destination AWS Region. Any automated snapshots copied to the destination AWS Region are deleted as they reach the retention period limit, unless you create manual snapshot copies of them. These manual snapshots, and any manual snapshots that were copied from the destination AWS Region, are kept in the destination AWS Region until you manually delete them.

To change the destination AWS Region that you copy snapshots to, first disable the automatic copy feature. Then re-enable it, specifying the new destination AWS Region.

After a snapshot is copied to the destination AWS Region, it becomes active and available for restoration purposes.

To copy snapshots for AWS KMS–encrypted clusters to another AWS Region, create a grant for Amazon Redshift to use a customer managed key in the destination AWS Region. Then choose that grant when you enable copying of snapshots in the source AWS Region. For more information about configuring snapshot copy grants, see Copying AWS KMS–encrypted snapshots to another AWS Region.

Restoring a cluster from a snapshot

A snapshot contains data from any databases that are running on your cluster. It also contains information about your cluster, including the number of nodes, node type, and admin user name. If you restore your cluster from a snapshot, Amazon Redshift uses the cluster information to create a new cluster. Then it restores all the databases from the snapshot data.

> **Note**
>
> No-backup tables aren't supported for RG or RA3 provisioned clusters and Amazon Redshift Serverless workgroups. A table marked as no-backup in an RG or RA3 cluster or serverless workgroup is treated as a permanent table that will always be backed up while taking a snapshot, and always restored when restoring from a snapshot.

For the new cluster created from the original snapshot, you can choose the configuration, such as node type and number of nodes. The cluster is restored in the same AWS Region and a random, system-chosen Availability Zone, unless you specify another Availability Zone in your request. When you restore a cluster from a snapshot, you can optionally choose a compatible maintenance track for the new cluster.

> **Note**
>
> When you restore a snapshot to a cluster with a different configuration, the snapshot must have been taken on a cluster with cluster version 1.0.10013, or later.

When a restore is in progress, events are typically emitted in the following order:

1. RESTORE_STARTED – REDSHIFT-EVENT-2008 sent when the restore process begins.
2. RESTORE_SUCCEEDED – REDSHIFT-EVENT-3003 sent when the new cluster has been created.

   The cluster is available for queries.
3. DATA_TRANSFER_COMPLETED – REDSHIFT-EVENT-3537 sent when data transfer complete.

> **Note**
>
> RG and RA3 clusters only emit RESTORE_STARTED and RESTORE_SUCCEEDED events. There is no explicit data transfer to be done after a RESTORE succeeds because RG and RA3 node types store data in Amazon Redshift managed storage. With RG and RA3 nodes, data is continuously transferred between RG and RA3 nodes and Amazon Redshift managed storage as part of normal query processing. RG and RA3 nodes cache hot data locally and keep less frequently queried blocks in Amazon Redshift managed storage automatically.

You can monitor the progress of a restore by either calling the DescribeClusters API operation, or viewing the cluster details in the AWS Management Console. For an in-progress restore, these display information such as the size of the snapshot data, the transfer rate, the elapsed time, and the estimated time remaining. For a description of these metrics, see RestoreStatus.

You can't use a snapshot to revert an active cluster to a previous state.

> **Note**
>
> When you restore a snapshot into a new cluster, the default security group and parameter group are used unless you specify different values.

You might want to restore a snapshot to a cluster with a different configuration for these reasons:

- When a cluster is made up of smaller node types and you want to consolidate it into a larger node type with fewer nodes.
- When you have monitored your workload and determined the need to move to a node type with more CPU and storage.
- When you want to measure performance of test workloads with different node types.

Restore has the following constraints:

- The new node configuration must have enough storage for existing data. Even when you add nodes, your new configuration might not have enough storage because of the way that data is redistributed.
- The restore operation checks if the snapshot was created on a cluster version that is compatible with the cluster version of the new cluster. If the new cluster has a version level that is too early, then the restore operation fails and reports more information in an error message.
- The possible configurations (number of nodes and node type) you can restore to is determined by the number of nodes in the original cluster and the target node type of the new cluster. To determine the possible configurations available, you can use the Amazon Redshift console or the `describe-node-configuration-options` AWS CLI command with `action-type restore-cluster`. For more information about the restoring using the Amazon Redshift console, see Restoring a cluster from a snapshot.

The following steps take a cluster with many nodes and consolidate it into a bigger node type with a smaller number of nodes using the AWS CLI. For this example, we start with a source cluster of 24 nodes. In this case, suppose that we already created a snapshot of this cluster and want to restore it into a bigger node type.

1. Run the following command to get the details of our 24-node cluster.

```
aws redshift describe-clusters --region eu-west-1 --cluster-identifier
 mycluster-123456789012
```

2. Run the following command to get the details of the snapshot.

```
aws redshift describe-cluster-snapshots --region eu-west-1 --snapshot-identifier
 mycluster-snapshot
```

3. Run the following command to describe the options available for this snapshot.

```
aws redshift describe-node-configuration-options --snapshot-identifier mycluster-
snapshot --region eu-west-1 --action-type restore-cluster
```

   This command returns an option list with recommended node types, number of nodes, and disk utilization for each option. For this example, the preceding command lists the following possible node configurations. We choose to restore into a four-node cluster.

```
{
    "NodeConfigurationOptionList": [
        {
            "NodeType": "ra3.16xlarge",
            "NumberOfNodes": 2,
            "EstimatedDiskUtilizationPercent": 0.1
        },
        {
            "NodeType": "ra3.16xlarge",
            "NumberOfNodes": 4,
            "EstimatedDiskUtilizationPercent": 0.05
        },
        {
            "NodeType": "ra3.4xlarge",
            "NumberOfNodes": 8,
            "EstimatedDiskUtilizationPercent": 0.03
        },
```

```
        {
            "NodeType": "ra3.4xlarge",
            "NumberOfNodes": 10,
            "EstimatedDiskUtilizationPercent": 0.02
        },
        {
            "NodeType": "ra3.4xlarge",
            "NumberOfNodes": 12,
            "EstimatedDiskUtilizationPercent": 0.02
        },
        {
            "NodeType": "ra3.xlplus",
            "NumberOfNodes": 8,
            "EstimatedDiskUtilizationPercent": 0.11
        },
        {
            "NodeType": "ra3.xlplus",
            "NumberOfNodes": 16,
            "EstimatedDiskUtilizationPercent": 0.06
        }
    ]
}
```

4. Run the following command to restore the snapshot into the cluster configuration that we chose. After this cluster is restored, we have the same content as the source cluster, but the data has been consolidated into four `ra3.16xlarge` nodes.

```
aws redshift restore-from-cluster-snapshot --region eu-west-1 --snapshot-identifier
 mycluster-snapshot --cluster-identifier mycluster-123456789012-x --node-type
 ra3.16xlarge --number-of-nodes 4
```

If you have reserved nodes, for example RA3 reserved nodes, you can upgrade to RG reserved nodes. You can do this when you restore from a snapshot or perform an elastic resize. You can use the console to guide you through this process. For more information about upgrading to RG nodes, see Upgrading to RG or RA3 node types.

To restore a cluster from a snapshot on the console

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, **Snapshots**, then choose the snapshot to restore.

3. Choose **Restore from snapshot** to view the **Cluster configuration** and **Cluster details** values of the new cluster to be created using the snapshot information.
4. Update the properties of the new cluster, then choose **Restore cluster from snapshot**.

After restoring your cluster snapshot, the restored data warehouse is encrypted with the same custom AWS KMS key that it was using at the time that the snapshot was taken. If the snapshot didn't have a custom KMS key, Amazon Redshift's backup encryption logic depends on the following factors:

- The type of Amazon Redshift data warehouse you're restoring the snapshot to.
- The encryption type of the cluster at the time the snapshot was taken.

To learn how your data warehouse is encrypted after you restore it from your cluster snapshot, see the following table:

Destination type	Snapshot encryption type	Destination encryption type
Provisioned cluster	Encrypted with an AWS managed key	Encrypted with an AWS managed key
Provisioned cluster	Encrypted with an AWS owned key	Encrypted with an AWS owned key
Serverless namespace	Encrypted with an AWS managed key	Encrypted with an AWS owned key
Serverless namespace	Encrypted with an AWS owned key	Encrypted with an AWS owned key

If AWS Secrets Manager managed your cluster's admin password at the time the snapshot was taken, you must continue using AWS Secrets Manager to manage the admin password. You can opt out of using a secret after restoring the cluster by updating the cluster's admin credentials in the cluster detail page.

If you have reserved nodes, you can upgrade to RG reserved nodes. You can do this when you restore from a snapshot or perform an elastic resize. You can use the console to guide you through

this process. For more information about upgrading to RG nodes, see Upgrading to RG or RA3 node types.

Restoring a table from a snapshot

You can restore a single table from a snapshot instead of restoring an entire cluster. When you restore a single table from a snapshot, you specify the source snapshot, database, schema, and table name, and the target database, schema, and a new table name for the restored table.

> **Note**
>
> No-backup tables aren't supported for RG or RA3 provisioned clusters and Amazon Redshift Serverless workgroups. A table marked as no-backup in an RG or RA3 cluster or serverless workgroup is treated as a permanent table that will always be backed up while taking a snapshot, and always restored when restoring from a snapshot. However, selective restoration of no-backup tables isn't supported.

The new table name cannot be the name of an existing table. To replace an existing table with a restored table from a snapshot, rename or drop the existing table before you restore the table from the snapshot.

The target table is created using the source table's column definitions, table attributes, and column attributes except for foreign keys. To prevent conflicts due to dependencies, the target table doesn't inherit foreign keys from the source table. Any dependencies, such as views or permissions granted on the source table, aren't applied to the target table.

If the owner of the source table exists, then that database user is the owner of the restored table, provided that the user has sufficient permissions to become the owner of a relation in the specified database and schema. Otherwise, the restored table is owned by the admin user that was created when the cluster was launched.

The restored table returns to the state it was in at the time the backup was taken. This includes transaction visibility rules defined by the Amazon Redshift adherence to serializable isolation, meaning that data will be immediately visible to in flight transactions started after the backup.

Restoring a table from a snapshot has the following limitations:

- You can restore a table only to the current, active running cluster and from a snapshot that was taken of that cluster.

- You can restore only one table at a time.
- You can't restore a table from a cluster snapshot that was taken prior to a cluster being resized. An exception is that you can restore a table after an elastic resize if the node type didn't change.
- Any dependencies, such as views or permissions granted on the source table, aren't applied to the target table.
- If row-level security is turned on for a table being restored, Amazon Redshift restores the table with row-level security turned on.

To restore a table from a snapshot

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the cluster that you want to use to restore a table.
3. For **Actions**, choose **Restore table** to display the **Restore table** page.
4. Enter the information about which snapshot, source table, and target table to use, and then choose **Restore table**.

Example: Restoring a table from a snapshot using the AWS CLI

The following example uses the `restore-table-from-cluster-snapshot` AWS CLI command to restore the `my-source-table` table from the `sample-database` schema in the `my-snapshot-id`. You can use the AWS CLI command `describe-table-restore-status` to review the status of your restore operation. The example restores the snapshot to the `mycluster-example` cluster with a new table name of `my-new-table`.

```
aws redshift restore-table-from-cluster-snapshot --cluster-identifier mycluster-
example
                                                 --new-table-name my-new-table
                                                 --snapshot-identifier my-snapshot-id
                                                 --source-database-name sample-
database
                                                 --source-table-name my-source-table
```

Restoring a serverless namespace from a snapshot

Restoring a serverless namespace from a snapshot replaces all of the namespace's databases with databases in the snapshot. For more information about serverless snapshots, see Snapshots and recovery points. Amazon Redshift automatically converts tables with interleaved keys into compound keys when you restore a provisioned cluster snapshot to an Amazon Redshift Serverless namespace. For more information about sort keys, see Working with sort keys.

To restore a snapshot from your provisioned cluster to your serverless namespace.

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, **Snapshots**, then choose the snapshot to use.
3. Choose **Restore from snapshot**, **Restore to serverless namespace**.
4. Choose the namespace you want to restore to.
5. Confirm you want to restore from your snapshot. Choose **restore**. This action replaces all the databases in serverless namespace with the data from your provisioned cluster.

Configuring cross-Region snapshot copy for a nonencrypted cluster

You can configure Amazon Redshift to copy snapshots for a cluster to another AWS Region. To configure cross-Region snapshot copy, you need to enable this copy feature for each cluster and configure where to copy snapshots and how long to keep copied automated or manual snapshots in the destination AWS Region. When cross-Region copy is enabled for a cluster, all new manual and automated snapshots are copied to the specified AWS Region. Copied snapshot names are prefixed with **`copy:`**.

To configure a cross-Region snapshot

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the cluster that you want to move snapshots for.
3. For **Actions**, choose **Configure cross-region snapshot**.

   The Configure cross-Region dialog box appears.
4. For **Copy snapshots**, choose **Yes**.

5. In **Destination AWS Region**, choose the AWS Region to which to copy snapshots.
6. In **Automated snapshot retention period (days)**, choose the number of days for which you want automated snapshots to be retained in the destination AWS Region before they are deleted.
7. In **Manual snapshot retention period**, choose the value that represents the number of days for which you want manual snapshots to be retained in the destination AWS Region before they are deleted. If you choose **Custom value**, the retention period must be between 1 to 3653 days.
8. Choose **Save**.

Configuring cross-Region snapshot copy for an AWS KMS–encrypted cluster

When you launch an Amazon Redshift cluster, you can configure a snapshot copy grant for a root key in your account in the destination AWS Region. If you don't configure a grant, snapshots in the destination region are encrypted with a default AWS-owned key. By doing this, you enable Amazon Redshift to perform encryption operations in the destination AWS Region.

The following procedure describes the process of enabling cross-Region snapshot copy for an AWS KMS-encrypted cluster. For more information about encryption in Amazon Redshift and snapshot copy grants, see Copying AWS KMS–encrypted snapshots to another AWS Region.

To configure a cross-Region snapshot for an AWS KMS–encrypted cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the cluster that you want to move snapshots for.
3. For **Actions**, choose **Configure cross-region snapshot**.

   The Configure cross-Region dialog box appears.
4. For **Copy snapshots**, choose **Yes**.
5. In **Destination AWS Region**, choose the AWS Region to which to copy snapshots.
6. In **Automated snapshot retention period (days)**, choose the number of days for which you want automated snapshots to be retained in the destination AWS Region before they are deleted.

7. In **Manual snapshot retention period**, choose the value that represents the number of days for which you want manual snapshots to be retained in the destination AWS Region before they are deleted. If you choose **Custom value**, the retention period must be between 1 to 3653 days.
8. Choose **Save**.

Modifying the manual snapshot retention period

You can change the retention period for a manual snapshot by modifying the snapshot settings.

To change the manual snapshot retention period

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, **Snapshots**, then choose the manual snapshot to change.
3. For **Actions**, choose **Manual snapshot settings** to display the properties of the manual snapshot.
4. Enter the revised properties of the snapshot definition, then choose **Save**.

Modifying the retention period for cross-Region snapshot copy

After you configure cross-Region snapshot copy, you might want to change the settings. You can easily change the retention period by selecting a new number of days and saving the changes.

> **Warning**
>
> You can't modify the destination AWS Region after cross-Region snapshot copy is configured.
> If you want to copy snapshots to a different AWS Region, first disable cross-Region snapshot copy. Then re-enable it with a new destination AWS Region and retention period. Any copied automated snapshots are deleted after you disable cross-Region snapshot copy. Thus, you should determine if there are any that you want to keep and copy them to manual snapshots before disabling cross-Region snapshot copy.

To modify a cross-Region snapshot

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the cluster that you want to modify snapshots for.
3. For **Actions**, choose **Configure cross-region snapshot** to display the properties of the snapshot.
4. Enter the revised properties of the snapshot definition, then choose **Save**.

Deleting a manual snapshot

You can delete manual snapshots by selecting one or more snapshots in the snapshot list.

To delete a manual snapshot

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, **Snapshots**, then choose the snapshot to delete.
3. For **Actions**, choose **Delete snapshot** to delete the snapshot.
4. Confirm the deletion of the listed snapshots, then choose **Delete**.

Registering a cluster to the AWS Glue Data Catalog

You can register entire clusters to the AWS Glue Data Catalog and create catalogs managed by AWS Glue. You can access these catalogs with any SQL engine that supports the Apache Iceberg REST API. For more information on creating Apache Iceberg-compatible catalogs from Amazon Redshift see Apache Iceberg compatibility for Amazon Redshift in the Amazon Redshift Database Developer Guide.

To register a cluster to the AWS Glue Data Catalog

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**. The clusters for your account in the current AWS Region are listed. A subset of properties of each cluster is displayed in columns in the list. If you don't have any clusters, choose **Create cluster** to create one.

3. Choose the name of the cluster that you want to register.
4. From **Actions**, choose **Register to AWS Glue Data Catalog**. The **Register to AWS Glue Data Catalog** pop-up box appears.
5. Enter the AWS account ID that you want to register the cluster to under **Destination account ID**. This is the account ID that will hold the catalog in the AWS Glue Data Catalog.
6. Enter a name under **Register namespace as**. This will be the cluster's name in the Data Catalog.
7. Choose **Register**. You'll be taken to the AWS Lake Formation console.
8. Follow the catalog creation process in AWS Lake Formation. For information about creating a catalog, see Bringing Amazon Redshift data into the AWS Glue Data Catalog in the AWS Lake Formation Developer Guide.

Multi-AZ deployment

Amazon Redshift supports multiple Availability Zones (Multi-AZ) deployments for provisioned RG or RA3 clusters. By using Multi-AZ deployments, your Amazon Redshift data warehouse can continue operating in failure scenarios when an unexpected event happens in an Availability Zone. A Multi-AZ deployment deploys compute resources in two Availability Zones (AZs) and these compute resources can be accessed through a single endpoint. In the event of an entire Availability Zone failure, the remaining compute resources in the second Availability Zone are available to continue processing workloads. Amazon Redshift charges the same hourly compute rates for RG or RA3 when running a Multi-AZ data warehouse. Storage costs remain the same as it is shared across all Availability Zones within and AWS Region.

Currently, Amazon Redshift supports zero Recovery Point Objective (RPO) that allows data to be current and up-to-date in the event of a failure. With Multi-AZ deployment, Amazon Redshift further enhances its existing recovery capabilities and reduces its Recovery Time Objective (RTO). This is possible because a Multi-AZ deployment can recover faster from a failure or disaster thereby elevating the Amazon Redshift Service Level Agreement (SLA) to 99.99% as compared to 99.9% with a Single-AZ data warehouse.

Setting up Multi-AZ deployment

To set up a Multi-AZ deployment, select the **Multi-AZ** option and specify the number of compute nodes to provision in each Availability Zone. Amazon Redshift automatically deploys equal

compute resources across two Availability Zones and all compute resources are always available for both read and write processing during normal operation and can opportunistically help to increase performance by using the extra compute resources in the secondary cluster. This allows a Multi-AZ deployment to act as a single data warehouse with a single endpoint, removing the need for application changes when a disaster occurs. Although a Multi-AZ deployment processes an individual query using the compute resources residing in only one Availability Zone, it can automatically distribute processing of some simultaneous queries to both Availability Zones that can sometimes, but not always, boost overall throughput for high concurrency workloads. It's important to keep in mind the main function of Multi-AZ, which is to provide a high-availability solution for your cluster to ensure rapid recovery from unplanned downtime or an outage event.

If your primary motivation is to obtain higher performance for your analytics workload, we recommend other solutions besides Multi-AZ, such as:

- resizing to a larger node instance type (Resizing clusters)
- enabling concurrency scaling on your cluster (Concurrency scaling)
- using data sharing architecture to isolate workloads (Data sharing overview)

You can also convert an existing Single-AZ data warehouse into a Multi-AZ data warehouse or vice versa. Everything remains the same except that additional compute resources are provisioned in the second Availability Zone. When migrating to Multi-AZ from an existing Single-AZ cluster, you might be required to double the number of cluster nodes needed, to facilitate that single query performance is maintained.

In the event of a failure in an Availability Zone, Amazon Redshift continues operating by using the resources in the remaining Availability Zone automatically. However, user connections might be lost and must be re-established. In addition, queries that were running in the failed Availability Zone can fail and must be retried. However, you can reconnect to your cluster and reschedule queries immediately, and Amazon Redshift will process the queries in the remaining Availability Zone. Queries issued at or after a failure occurs might experience runtime delays while the Multi-AZ data warehouse is recovering.

Note

To achieve better performance and higher availability, we recommend that you use SNAPSHOT ISOLATION with your Multi-AZ clusters. For more information, see CREATE DATABASE.

Limitations

A Multi-AZ data warehouse has the same functional capabilities as a Single-AZ data warehouse, except for the following limitations that apply to a Multi-AZ data warehouse:

- You can't create an unencrypted Multi-AZ data warehouse. Make sure to add an encryption when creating a new Multi-AZ data warehouse, converting a Single-AZ data warehouse into a Multi-AZ data warehouse, or converting a Single-AZ data warehouse into a Multi-AZ data warehouse.
- You can't create a single node Multi-AZ deployment for any of the RG or RA3 instance types. Choose 2 or more nodes per Availability Zone while creating a Multi-AZ deployment.
- Amazon Redshift doesn't support a subnet configuration that can support fewer than three Availability Zones. In other words, the configured subnet group requires three or more subnets.
- You can't relocate a Multi-AZ deployment to another Availability Zone. Relocation will be automatically determined and conducted by Amazon Redshift when using Multi-AZ deployment.
- You can't pause or resume a Multi-AZ deployment.
- You can't run your Multi-AZ deployment outside of the supported port ranges 5431 to 5455 and 8191 to 8215.
- You can't use STL, SVCS, SVL, SVV, STV views with Multi-AZ deployments as they only support system monitoring views (SYS_* views). Change your monitoring queries to use system monitoring views (SYS_* views).
- You can't attach an Elastic IP address to an existing cluster with Multi-AZ enabled.
- You can't convert a cluster with an attached Elastic IP address from Single-AZ to Multi-AZ.
- Amazon Redshift Multi-AZ deployment is available in the following AWS Regions:

> **Note**
>
> RG instances are available in select regions. Refer to RG node type availability in AWS Regions.

  - US East (Ohio) (us-east-2)
  - US East (N. Virginia) (us-east-1)
  - US West (Oregon) (us-west-2)
  - Africa (Cape Town) (af-south-1)
  - Asia Pacific (Hong Kong) (ap-east-1)

- Asia Pacific (Taipei) (ap-east-2)
- Asia Pacific (Hyderabad) (ap-south-2)
- Asia Pacific (Jakarta) (ap-southeast-3)
- Asia Pacific (Malaysia) (ap-southeast-5)
- Asia Pacific (Melbourne) (ap-southeast-4)
- Asia Pacific (Mumbai) (ap-south-1)
- Asia Pacific (Osaka) (ap-northeast-3)
- Asia Pacific (Seoul) (ap-northeast-2)
- Asia Pacific (Singapore) (ap-southeast-1)
- Asia Pacific (Sydney) (ap-southeast-2)
- Asia Pacific (New Zealand) (ap-southeast-6)
- Asia Pacific (Thailand) (ap-southeast-7)
- Asia Pacific (Tokyo) (ap-northeast-1)
- Canada (Central) (ca-central-1)
- China (Beijing) (cn-north-1)
- China (Ningxia) (cn-northwest-1)
- Europe (Frankfurt) (eu-central-1)
- Europe (Ireland) (eu-west-1)
- Europe (London) (eu-west-2)
- Europe (Milan) (eu-south-1)
- Europe (Paris) (eu-west-3)
- Europe (Spain) (eu-south-2)
- Europe (Stockholm) (eu-north-1)
- Europe (Zurich) (eu-central-2)
- Israel (Tel Aviv) (il-central-1)
- Mexico (Central) (mx-central-1)
- Middle East (Bahrain) (me-south-1)
- Middle East (UAE) (me-central-1)
- South America (São Paulo) (sa-east-1)

- AWS GovCloud (US-East) (us-gov-east-1)

- AWS GovCloud (US-West) (us-gov-west-1)
- Publicly accessible Multi-AZ data warehouses support 1 less VPC security group than Single-AZ and privately accessible Multi-AZ warehouses.

Setting up Multi-AZ when creating a new cluster

Amazon Redshift Multi-AZ supports two Availability Zones at a time. Amazon Redshift automatically selects the Availability Zones based on the selected subnet group configuration. You can convert an existing single Availability Zone data warehouse into a Multi-AZ data warehouse or restore from a snapshot to configure it into a Multi-AZ data warehouse.

Using the Amazon Redshift console, you can easily create new Multi-AZ deployments. To create a new Multi-AZ deployment using the Amazon Redshift console, select the Multi-AZ option when creating the data warehouse. Specify the number of compute nodes required in a single Availability Zone, and Amazon Redshift will deploy that number of nodes in each of two Availability Zones. All nodes will be used to read and write workload processing during normal operation. You can also use the AWS CLI `create-cluster` command to create a new Multi-AZ data warehouse using the `multi-az` parameter.

You can convert an existing Single-AZ data warehouse into a Multi-AZ data warehouse, you can use either the Amazon Redshift console or the AWS CLI `modify-cluster` command using the `multi-az` parameter. Or, you can restore from a snapshot to configure a Single-AZ data warehouse into a Multi-AZ data warehouse either using the Amazon Redshift console or the AWS CLI `restore-from-cluster-snapshot` command using the `multi-az` parameter.

Multi-AZ deployment only supports RG or RA3 node types that use Amazon Redshift Managed Storage (RMS). Amazon Redshift stores data in RMS, which uses Amazon S3 and is accessible in all Availability Zones in an AWS Region, without having to replicate the data at the Amazon Redshift level.

You can set up Multi-AZ deployment when creating a new cluster either using the Amazon Redshift console or the AWS Command Line Interface.

Using the console

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.

2. On the navigation menu, choose **Provisioned clusters dashboard**, and choose **Clusters**. The clusters for your account in the current AWS Region are listed. A subset of properties of each cluster is displayed in columns in the list.
3. Choose the button **Create cluster** to open the create cluster page.
4. Enter properties for your cluster. For general information about creating clusters, see Creating a cluster.
5. Choose one of the RG or RA3 node types from the **Node type** drop-down list. The AZ configuration option becomes available only when you chose an RG or RA3 node type.
6. Under **AZ configuration**, choose **Multi-AZ**.
7. Under **Number of nodes per AZ**, enter at least two nodes for your cluster.
8. You have the option to load sample data or bring your own data:

   - In **Sample data**, choose **Load sample data** to load the sample dataset into your Amazon Redshift cluster. Amazon Redshift loads the sample dataset Tickit into the default dev database and public schema. Amazon Redshift automatically loads the sample dataset into your Amazon Redshift cluster. You can start using the query editor v2 to query data.
   - To bring your own data to your Amazon Redshift cluster, follow the steps in Bringing your own data to Amazon Redshift.
9. Scroll down to **Additional configurations**, expand **Network and security**, and make sure that you either accept the default **Cluster subnet group** or choose another one. If you choose another cluster subnet group, make sure that there are 3 Availability Zones in the subnet group you selected.
10. Under **Additional configurations**, expand **Database configurations**.
11. To use a custom AWS KMS key instead of the default AWS-owned key, click **Customize encryption settings** under **Database encryption**.
12. Under **Choose an KMS key**, you can either choose an AWS Key Management Service key or enter an ARN. Or, you can click **Create an AWS Key Management Service key** in the AWS Key Management Service console. For more information about creating KMS keys, see Creating Keys in the *AWS Key Management Service Developer Guide*.
13. Click **Create cluster**. When the cluster creation succeeds, you can view the details in the cluster details page. You can use your SQL client to load and query data.

Using the AWS Command Line Interface

To set up Multi-AZ when creating a cluster using the AWS Command Line Interface

- From the AWS CLI use the `create-cluster` command and the `multi-az` parameter as follows.

```
aws redshift create-cluster
    --port 5439
    --master-username master
    --master-user-password #####
    --node-type rg.4xlarge
    --number-of-nodes 2
    --profile maz-test
    --endpoint-url https://redshift.eu-west-1.amazonaws.com
    --region eu-west-1
    --cluster-identifier test-maz
    --multi-az
    --maintenance-track-name CURRENT
    --encrypted
```

Setting up Multi-AZ for a data warehouse restored from a snapshot

To create a new Multi-AZ cluster by restoring it from a snapshot, complete the following procedure.

Using the console

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, **Snapshots**, then choose the snapshot to use.
3. Choose **Restore snapshot**, **Restore to a provisioned cluster**.
4. Enter properties for your cluster. For general information about creating clusters, see Creating a cluster.
5. Choose one of the RG or RA3 node types from the **Node type** drop-down list. The AZ configuration option becomes available only when you chose an RG or RA3 node type.
6. Under **AZ configuration**, choose **Multi-AZ**.
7. Under **Number of nodes per AZ**, enter at least two nodes for your cluster.
8. You have the option to load sample data or bring your own data:

- In **Sample data**, choose **Load sample data** to load the sample dataset into your Amazon Redshift cluster. Amazon Redshift loads the sample dataset Tickit into the default dev database and public schema. Amazon Redshift automatically loads the sample dataset into your Amazon Redshift cluster. You can start using the query editor v2 to query data.
- To bring your own data to your Amazon Redshift cluster, follow the steps in Load data from Amazon S3 to Amazon Redshift.

9. Scroll down to **Additional configurations**, expand **Network and security**, and make sure that you either accept the default **Cluster subnet group** or choose another one. If you choose another cluster subnet group, make sure that there are 3 Availability Zones in the subnet group you selected.
10. Under **Additional configurations**, expand **Database configurations**.
11. Under **Database encryption**, to use a custom KMS key other than the default AWS-owned key, click **Customize encryption settings**. This option is deselected by default.
12. Under **Choose an KMS key**, you can either choose an AWS Key Management Service key or enter an ARN. Or, you can click **Create an AWS Key Management Service key** in the AWS Key Management Service console. For more information about creating KMS keys, see Creating Keys in the *AWS Key Management Service Developer Guide*.
13. Click **Restore cluster from snapshot**. When the cluster restoration succeeds, you can view the details in the cluster details page.

Using the AWS Command Line Interface

- From the AWS CLI, use the `restore-from-cluster-snapshot` command as follows.

```
aws redshift restore-from-cluster-snapshot
--region eu-west-1
--multi-az
--snapshot-identifier test-snap1
--cluster-identifier test-saz-11
--endpoint-url https://redshift.eu-west-1.amazonaws.com/
```

Converting a Single-AZ data warehouse to a Multi-AZ data warehouse

By converting a Single-AZ data warehouse to a Multi-AZ data warehouse, your data warehouse will be highly available with 99.99% SLA guarantee. The performance of an individual query will

remain same even with a Multi-AZ data warehouse. For higher concurrency workloads, you will see a boost in overall throughput as Amazon Redshift can execute requests using compute resources in two Availability Zones.

> **Note**
>
> Amazon Redshift will not allow you to split existing compute resources while converting from Single-AZ to Multi-AZ, or vice versa. This operation isn't supported to maintain consistent individual query performance.

Using the console

To convert a Single-AZ cluster to a Multi-AZ data warehouse using the console

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Provisioned clusters dashboard**, and choose **Clusters**. The clusters for your account in the current AWS Region are listed. A subset of properties of each cluster is displayed in columns in the list.
3. Choose the cluster you want to convert to a Multi-AZ deployment. The cluster details page appears.
4. For **Actions**, choose **Activate Multi-AZ**. The modification summary appears. Click **Activate Multi-AZ**.
5. When there is an error, do one of the following, then click **Activate Multi-AZ**.
   - Cluster encryption — Choose **Properties** to edit the encryption settings in the Database configuration section under the Properties tab of the cluster details page.
   - Subnet group — Choose **Subnet group** to edit the cluster subnet group settings by clicking the subnet group link. If you choose another cluster subnet group, make sure that there are 3 Availability Zones in the subnet group you selected.
   - Port settings — Choose **Properties** to edit the port setting in the Database configuration section under the Properties tab of the cluster details page.
6. You can use your SQL client to load and query data.

Using the AWS Command Line Interface

- From the AWS CLI, use the `modify-cluster` command and the `multi-az` parameter as follows.

```
aws redshift modify-cluster
    --profile maz-test
    --endpoint-url https://redshift.eu-west-1.amazonaws.com
    --region eu-west-1
    --cluster-identifier test-maz-11
    --multi-az
```

You can't use STL, SVCS, SVL, SVV, or STV views with Multi-AZ deployments because they only support system monitoring views (SYS_* views). Change your monitoring queries to use system monitoring views (SYS_* views).

Converting a Multi-AZ data warehouse to a Single-AZ data warehouse

By converting a Multi-AZ data warehouse to a Single-AZ data warehouse, your data warehouse will not get the 99.99% SLA guarantee which Multi-AZ offers. The performance of an individual query will remain same but the overall throughput will be affected because compute resources in the second Availability Zone won't be available. You have the option to enable concurrency scaling to automatically scale throughput for consistent performance even with Single-AZ.

> **Note**
>
> Amazon Redshift will not allow you to split existing compute resources while converting from Single-AZ to Multi-AZ, or vice versa. This operation isn't supported to maintain consistent individual query performance.

Using the console

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Provisioned clusters dashboard**, and choose **Clusters**. The clusters for your account in the current AWS Region are listed. A subset of properties of each cluster is displayed in columns in the list.

3. Choose the cluster you want to convert to a Multi-AZ deployment. The cluster details page appears.
4. For **Actions**, choose **Deactivate Multi-AZ**. The modification summary appears. Click **Deactivate Multi-AZ**.

Using the AWS Command Line Interface

- From the AWS CLI, use the `modify-cluster` command and the `no-multi-az` parameter as follows.

```
aws redshift modify-cluster
    --profile maz-test
    --endpoint-url  https://redshift.eu-west-1.amazonaws.com
    --region eu-west-1
    --cluster-identifier test-maz-11
    --no-multi-az
```

Once your data warehouse is converted to Single-AZ, it will lose the 99.99 SLA guarantee. Overall throughput will also be impacted. When the changes are saved, you can view the details in the cluster details page.

Resizing a Multi-AZ data warehouse

You can resize a Multi-AZ data warehouse and specify a number of nodes or node type that is different from the current configuration of the data warehouse.

Using the console

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Provisioned clusters dashboard**, and choose **Clusters**. The clusters for your account in the current AWS Region are listed. A subset of properties of each cluster is displayed in columns in the list.
3. Choose the cluster you want to resize the Multi-AZ data warehouse. The cluster details page appears.
4. For **Actions**, choose **Resize**. The Resize cluster page appears.

5. Follow the instructions on the page. You can resize the cluster now, once at a specific time, or increase and decrease the size of your cluster on a schedule.
6. Under **New configurations**, choose one of the RG or RA3 node types from the Node type drop-down list.
7. Click **Resize cluster**.

Using the AWS Command Line Interface

To resize a Multi-AZ data warehouse using the AWS Command Line Interface

- From the AWS CLI, use the `resize-cluster` command to change the number of nodes for a single Availability Zone as follows.

```
aws redshift resize-cluster \
    --cluster-identifier test-maz-11
    --cluster-type multi-node
    --node-type rg.4xlarge
    --number-of-nodes 6
```

Failing over Multi-AZ deployment

Your Multi-AZ data warehouse is a collection compute resources deployed simultaneously in two Availability Zones. The compute resources deployed in the primary Availability Zone are referred to as primary compute and those in the secondary Availability Zones are referred as secondary compute. A Multi-AZ data warehouse can automatically recover without any user intervention during an unlikely event such as an Availability Zone or infrastructure failure. The recovery process involves failing over from primary compute to secondary compute and designating secondary compute resources as primary. Additionally, new secondary compute resources are provisioned in a third Availability Zone. The automatic recovery process is measured in terms of RTO and RPO.

- **Recovery time objective (RTO)** — The time it takes a system to return to a working state after a disaster. In other words, RTO measures downtime.
- **Recovery point objective (RPO)** — The amount of data that can be lost (measured in time). For an Amazon Redshift Multi-AZ data warehouse, RPO is typically is zero as all the data is stored in Amazon Redshift Managed Storage (RMS), backed by Amazon Simple Storage Service, which is a highly durable and available by default.

> **Note**
>
> The performance of an individual query performance will not change after a failover has occurred. The overall throughput of your data warehouse will be reduced for a short time due to unavailability of compute resources in one of the Availability Zones. However, Amazon Redshift will automatically acquire capacity in another Availability Zone to ensure the same data warehouse processing capacity is restored.

In addition to the automatic recovery process, you can also trigger this process manually for your data warehouse using the **Failover primary compute** option. You can use this approach to test how Multi-AZ would help your application for higher high availability and better continuity.

Using the console

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. Do one of the following:
   - On the navigation menu, choose **Clusters**. Under **Clusters**, choose a cluster. The cluster details page appears.
   - From the cluster dashboard, choose a cluster.
3. From **Actions**, choose **Failover primary compute**.
4. When prompted, click **Confirm**.

Using the AWS Command Line Interface

- From the AWS CLI, use the `failover-primary-compute` command as follows.

```
aws redshift failover-primary-compute
    --profile maz-test
    --endpoint-url  https://redshift.eu-west-1.amazonaws.com
    --region eu-west-1
    --cluster-identifier test-maz-11
```

After the above operation is confirmed, Amazon Redshift will perform the same steps as an automatic recovery from an Availability Zone or infrastructure failure. The process will cause

compute nodes in the primary Availability Zone to become unavailable and compute resources in the secondary Availability Zone will be designated as primary compute. When the cluster recovery successfully completes, Multi-AZ deployment becomes available. Your Multi-AZ data warehouse will also automatically provision new secondary compute in another third Availability Zone as soon as it is available.

During this process, the cluster status on the console shows as modifying for the entire time, as the cluster automatically recovers and reconfigures back to the Multi-AZ deployment setup. The cluster can accept new connections immediately. Existing connections and inflight queries might be dropped. You can retry them immediately.

Viewing queries and loads for Multi-AZ data warehouses

You can view information on queries that ran in the past 7 days irrespective of the type, size, and status (pause or resume) of your cluster.

The information shown on the queries and loads page is populated with information from Amazon Redshift system tables (SYS_* views). This information lets you display additional information about your queries and offers rolling 7 days of retention. Query diagnostics become faster, letting you filter data by database, username, or SQL statement type. To see these additional filters and information on all queries that ran, note the following prerequisites:

- You must connect to a database by choosing **Connect to database**.
- Your database user must have the sys:operator or sys:monitor roles and permissions to perform query monitoring. For information about system roles, see Amazon Redshift system-defined roles in the *Amazon Redshift Database Developer Guide*.

You will see these additional filters and query information once you connect to a database.

To display query performance data from Queries and loads

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Queries and loads** to display the list of queries for your account.
3. You might have to connect to a database to see additional filter. If required, click **Connect to database** and follow the prompts to connect to a database.

By default, the list displays queries for all your clusters over the past 24 hours. You can change the scope of the displayed date in the console.

To display query performance data from Query monitoring

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**. Under **Clusters**, select a cluster.
3. Choose **Query monitoring**.
4. Depending on the configuration or version of your cluster, you might have to connect to a database to see additional filters. If required, click **Connect to database** and follow the prompts to connect to a database.

Monitoring a query in a Multi-AZ deployment

A Multi-AZ deployment uses compute resources that are deployed in both Availability Zones and can continue operating in the event that the resources in a given Availability Zone aren't available. All the compute resources will be used at all times. This allows full operation across two Availability Zones in an active-active fashion for both read and write operations.

You can query SYS_ views in pg_catalog schema to monitor query runtime in a Multi-AZ deployment. The SYS_ views display query runtime activities or statistics from primary and secondary clusters. For a list of monitoring views, see Monitoring views.

Follow these steps to monitor query runtime for each Availability Zone within the Multi-AZ deployment:

1. Navigate to the Amazon Redshift console and connect to the database in your Multi-AZ deployment and run queries through the query editor.
2. Run any sample query on the Multi-AZ Amazon Redshift deployment.
3. For a Multi-AZ deployment, you can identify a query and the Availability Zone where it is run by using the compute_type column in the SYS_QUERY_HISTORY table. *primary* stands for queries run on the primary cluster in the Multi-AZ deployment, and *secondary* stands for queries run on the secondary cluster in the Multi-AZ deployment.

   The following query uses compute_type column to monitor a query.

```
select (compute_type) as compute_type, left(query_text, 50) query_text from
 sys_query_history order by start_time desc;

 compute_type | query_text
--------------+------------------------
   secondary  | select count(*) from t1;
```

Terminating a query for a cluster

The procedure is applicable to both Multi-AZ and Single-AZ clusters.

To terminate a query

You can also use the **Queries** page to end a query that is currently in progress.

Your database user must have the sys:operator role and permissions to end a running query. For information about system roles, see Amazon Redshift system-defined roles in the *Amazon Redshift Database Developer Guide*.

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Queries and loads** to display the list of queries for your account.
3. Choose the running query that you want to end in the list, and then choose **Terminate query**.

Monitoring Amazon Redshift cluster performance

Amazon Redshift provides performance metrics and data so that you can track the health and performance of your clusters and databases. In this section, we discuss the types of data that you can work with in Amazon Redshift, specifically in the Amazon Redshift console.

The performance data that you can use in the Amazon Redshift console falls into two categories:

- **Amazon CloudWatch metrics –** Amazon CloudWatch metrics help you monitor physical aspects of your cluster, such as CPU utilization, latency, and throughput. Metric data is displayed directly in the Amazon Redshift console. You can also view it in the CloudWatch console. Alternatively,

you can consume it in any other way you work with metrics, such as with the AWS CLI or one of the AWS SDKs.

- **Query/Load performance data –** Performance data helps you monitor database activity and performance. This data is aggregated in the Amazon Redshift console to help you easily correlate what you see in CloudWatch metrics with specific database query and load events. You can also create your own custom performance queries and run them directly on the database. Query and load performance data is displayed only in the Amazon Redshift console. It is not published as CloudWatch metrics.

Performance data is integrated into the Amazon Redshift console, yielding a richer experience in the following ways:

- Performance data associated with a cluster is displayed contextually when you view a cluster, where you might need it to make decisions about the cluster such as resizing.
- Some performance metrics are displayed in more appropriately scaled units in the Amazon Redshift console as compared to CloudWatch. For example, `WriteThroughput`, is displayed in GB/s (as compared to bytes/s in CloudWatch), which is a more relevant unit for the typical storage space of a node.
- You can easily display performance data for the nodes of a cluster together on the same graph. This way, you can easily monitor the performance of all nodes of a cluster. You can also view performance data for each node.

Amazon Redshift provides performance data (both CloudWatch metrics and query and load data) at no additional charge. Performance data is recorded every minute. You can access historical values of performance data in the Amazon Redshift console. For detailed information about using CloudWatch to access the Amazon Redshift performance data that is exposed as CloudWatch metrics, see What is CloudWatch? in the *Amazon CloudWatch User Guide*.

Performance data in Amazon Redshift

Using CloudWatch metrics for Amazon Redshift, you can get information about your cluster's health and performance and see information at the node level. When working with these metrics, keep in mind that each metric has one or more dimensions associated with it. These dimensions tell you what the metric is applicable to, that is the scope of the metric. Amazon Redshift has the following two dimensions:

- Metrics that have a `NodeID` dimension are metrics that provide performance data for nodes of a cluster. This set of metrics includes leader and compute nodes. Examples of these metrics include `CPUUtilization`, `ReadIOPS`, `WriteIOPS`.
- Metrics that have only a `ClusterIdentifier` dimension are metrics that provide performance data for clusters. Examples of these metrics include `HealthStatus` and `MaintenanceMode`.

> **Note**
>
> In some metric cases, a cluster-specific metric represents an aggregation of node behavior. In these cases, take care in the interpretation of the metric value because the leader node's behavior is aggregated with the compute node.

For general information about CloudWatch metrics and dimensions, see CloudWatch concepts in the *Amazon CloudWatch User Guide*.

For a further description of CloudWatch metrics for Amazon Redshift, see the following sections.

Topics

- Amazon Redshift metrics
- Dimensions for Amazon Redshift metrics
- Amazon Redshift query and load performance data

Amazon Redshift metrics

The `AWS/Redshift` namespace includes the following metrics. Unless stated otherwise, metrics are collected at 1-minute intervals.

Metric	Description
`CommitQueueLength`	The number of transactions waiting to commit at a given point in time. Units: Count Dimensions: `ClusterIdentifier`

Metric	Description
ConcurrencyScaling ActiveClusters	The number of concurrency scaling clusters that are actively processing queries at any given time. Units: Count Dimensions: ClusterIdentifier
ConcurrencyScaling Seconds	The number of seconds used by concurrency scaling clusters that have active query processing activity. Units: Count Dimensions: ClusterIdentifier
CPUUtilization	The percentage of CPU utilization. For clusters, this metric represents an aggregation of all nodes (leader and compute) CPU utilization values. Units: Percent Dimensions: ClusterIdentifier , NodeID Dimensions: ClusterIdentifier
DatabaseConnections	The number of database connections to a cluster. Units: Count Dimensions: ClusterIdentifier

Metric	Description
HealthStatus	Indicates the health of the cluster. Every minute the cluster connects to its database and performs a simple query. If it is able to perform this operation successfully, the cluster is considered healthy. Otherwise, the cluster is unhealthy. An unhealthy status can occur when the cluster database is under extremely heavy load or if there is a configuration problem with a database on the cluster. **Note** In Amazon CloudWatch, this metric is reported as 1 or 0 whereas in the Amazon Redshift console, this metric is displayed with the words HEALTHY or UNHEALTHY for convenience. When this metric is displayed in the Amazon Redshift console, sampling averages are ignored and only HEALTHY or UNHEALTHY are displayed. In Amazon CloudWatch, values different than 1 and 0 might occur because of sampling issue. Any value below 1 for HealthStatus is reported as 0 (UNHEALTHY). Units: Count (1/0) (HEALTHY/UNHEALTHY in the Amazon Redshift console) Dimensions: ClusterIdentifier

Metric	Description
MaintenanceMode	Indicates whether the cluster is in maintenance mode. **Note** In Amazon CloudWatch, this metric is reported as 1 or 0 whereas in the Amazon Redshift console, this metric is displayed with the words ON or OFF for convenience. When this metric is displayed in the Amazon Redshift console, sampling averages are ignored and only ON or OFF are displayed. In Amazon CloudWatch, values different than 1 and 0 might occur because of sampling issues. Any value greater than 0 for MaintenanceMode is reported as 1 (ON). Units: Count (1/0) (ON/OFF in the Amazon Redshift console). Dimensions: ClusterIdentifier
MaxConfiguredConcurrencyScalingClusters	Maximum number of concurrency scaling clusters configured from the parameter group. For more information, see Amazon Redshift parameter groups. Units: Count Dimensions: ClusterIdentifier
NetworkReceiveThroughput	The rate at which the node or cluster receives data. Units: Bytes/Second (MB/s in the Amazon Redshift console) Dimensions: ClusterIdentifier , NodeID Dimensions: ClusterIdentifier

Metric	Description
NetworkTransmitThroughput	The rate at which the node or cluster writes data. Units: Bytes/Second (MB/s in the Amazon Redshift console) Dimensions: ClusterIdentifier , NodeID Dimensions: ClusterIdentifier
PercentageDiskSpaceUsed	The percent of disk space used. Units: Percent Dimensions: ClusterIdentifier Dimensions: ClusterIdentifier , NodeID
QueriesCompletedPerSecond	The average number of queries completed per second. Reported in 5-minute intervals. This metric isn't supported on single-node clusters. Units: Count/Second Dimensions: ClusterIdentifier , latency Dimensions: ClusterIdentifier , wlmid
QueryDuration	The average amount of time to complete a query. Reported in 5-minute intervals. This metric isn't supported on single-node clusters. Units: Microseconds Dimensions: ClusterIdentifier , NodeID, latency Dimensions: ClusterIdentifier , latency Dimensions: ClusterIdentifier , NodeID, wlmid

Metric	Description
QueryRuntimeBreakdown	The total time queries spent running by query stage. Reported in 5-minute intervals. Units: Milliseconds Dimensions: ClusterIdentifier, NodeID, stage Dimensions: ClusterIdentifier, stage
ReadIOPS	The average number of disk read operations per second. Units: Count/Second Dimensions: ClusterIdentifier , NodeID Dimensions: ClusterIdentifier
ReadLatency	The average amount of time taken for disk read I/O operations. Units: Seconds Dimensions: ClusterIdentifier , NodeID Dimensions: ClusterIdentifier
ReadThroughput	The average number of bytes read from disk per second. Units: Bytes (GB/s in the Amazon Redshift console) Dimensions: ClusterIdentifier , NodeID Dimensions: ClusterIdentifier
RedshiftManagedStorageTotalCapacity	Total managed storage capacity. Units: Megabytes Dimensions: ClusterIdentifier

Metric	Description
RgUniqueSnapshotStorage	The number of megabytes of unique data blocks stored across all active manual snapshots for the RG instance family. Units: Megabytes
TotalTableCount	The total number of tables at a particular point in time. This total includes permanent tables, temporary tables, datashare tables, external tables, and materialized views. Views and system tables are not included in this count. Units: Count Dimensions: ClusterIdentifier
WLMQueueLength	The number of queries waiting to enter a workload management (WLM) queue. Units: Count Dimensions: ClusterIdentifier , service class Dimensions: ClusterIdentifier , QueueName
WLMQueueWaitTime	The total time queries spent waiting in the workload management (WLM) queue. Reported in 5-minute intervals. Units: Milliseconds. Dimensions: ClusterIdentifier , QueryPriority Dimensions: ClusterIdentifier , wlmid Dimensions: ClusterIdentifier , QueueName

Metric	Description
WLMQueriesCompletedPerSecond	The average number of queries completed per second for a workload management (WLM) queue. Reported in 5-minute intervals. This metric isn't supported on single-node clusters. Units: Count/Second Dimensions: ClusterIdentifier , wlmid Dimensions: ClusterIdentifier , QueueName
WLMQueryDuration	The average length of time to complete a query for a workload management (WLM) queue. Reported in 5-minute intervals. This metric isn't supported on single-node clusters. Units: Microseconds Dimensions: ClusterIdentifier , wlmid Dimensions: ClusterIdentifier , QueueName
WLMRunningQueries	The number of queries running from both the main cluster and concurrency scaling cluster per WLM queue. Units: Count Dimensions: ClusterIdentifier , wlmid Dimensions: ClusterIdentifier , QueueName
WriteIOPS	The average number of write operations per second. Units: Count/Second Dimensions: ClusterIdentifier , NodeID Dimensions: ClusterIdentifier

Metric	Description
WriteLatency	The average amount of time taken for disk write I/O operations. Units: Seconds Dimensions: ClusterIdentifier , NodeID Dimensions: ClusterIdentifier
WriteThroughput	The average number of bytes written to disk per second. Units: Bytes (GB/s in the Amazon Redshift console) Dimensions: ClusterIdentifier , NodeID Dimensions: ClusterIdentifier
SchemaQuota	The configured quota for a schema. Units: Megabytes Dimensions: ClusterIdentifier , Database, Schema Periodic/Push: Periodic Frequency: 5 minutes Stop criteria: Schema dropped or quota removed
NumExceededSchemaQuotas	The number of schemas with exceeded quotas. Units: Count Dimensions: ClusterIdentifier Periodic/Push: Periodic Frequency: 5 minutes Stop criteria: N/A

Metric	Description
StorageUsed	The disk or storage space used by a schema. Units: Megabytes Dimensions: ClusterIdentifier , Database, Schema Periodic/Push: Periodic Frequency: 5 minutes Stop criteria: Schema dropped or quota removed
PercentageQuotaUsed	The percentage of disk or storage space used relative to the configured schema quota. Units: Percent Dimensions: ClusterIdentifier , Database, Schema Periodic/Push: Periodic Frequency: 5 minutes Stop criteria: Schema dropped or quota removed

Metric	Description
UsageLimitAvailable	Depending on the FeatureType, UsageLimitAvailable returns the following: • If the FeatureType is CONCURRENCY_SCALING , UsageLimitAvailable returns the total amount of time that can be used by concurrency scaling in 1-minute increments. This time doesn't count as time available for CONCURRENCY_SCALING_AUTO_TASK and vice versa. • If the FeatureType is CONCURRENCY_SCALING_AUTO_TASK , UsageLimitAvailable returns the total amount of time that can be used by concurrency scaling for automatic optimization tasks. This time doesn't count as time available for the CONCURRENCY_SCALING and vice versa. • If the FeatureType is EXTRA_COMPUTE_FOR_AUTOMATIC_OPTIMIZATION , UsageLimitAvailable returns the total amount of time available to use for automatic optimization tasks. • If the FeatureType is CROSS_REGION_DATASHARING , UsageLimitAvailable returns the total amount of data that can be scanned in 1-TB increments. • If the FeatureType is SPECTRUM, UsageLimitAvailable returns the total amount of data that can be scanned in 1-TB increments. Units: Minutes or TBs Dimensions: ClusterIdentifier , FeatureType , UsageLimitId

Metric	Description
UsageLimitConsumed	Depending on the FeatureType, UsageLimitConsumed returns the following: • If the FeatureType is CONCURRENCY_SCALING , UsageLimitConsumed returns the total amount of time used by concurrency scaling in 1-minute increments. This time doesn't count as time used for CONCURRENCY_SCALING_AUTO_TASK and vice versa. • If the FeatureType is CONCURRENCY_SCALING_AUTO_TASK , UsageLimitConsumed returns the total amount of time that was used by concurrency scaling for automatic optimization tasks. This time doesn't count as time used for CONCURRENCY_SCALING and vice versa. • If the FeatureType is EXTRA_COMPUTE_FOR_AUTOMATIC_OPTIMIZATION , UsageLimitConsumed returns the total amount of time used for automatic optimization tasks. • If the FeatureType is CROSS_REGION_DATASHARING , UsageLimitConsumed returns the total amount of data scanned in 1-TB increments. • If the FeatureType is SPECTRUM, UsageLimitConsumed returns the total amount of data scanned in 1-TB increments. Units: Minutes or TBs Dimensions: ClusterIdentifier , FeatureType , UsageLimitId

Dimensions for Amazon Redshift metrics

Amazon Redshift data can be filtered along any of the dimensions in the table following.

Dimension	Description
latency	Possible values are as follows: • short – under 10 seconds • medium – between 10 seconds and 10 minutes • long – over 10 minutes
NodeID	Filters requested data that is specific to the nodes of a cluster. NodeID is either "Leader", "Shared", or "Compute-N" where N is 0, 1, ... for the number of nodes in the cluster. "Shared" means that the cluster has only one node, that is the leader node and compute node are combined. Metrics are reported for the leader node and compute nodes only for CPUUtilization , NetworkTransmitThroughput , and ReadIOPS. Other metrics that use the NodeId dimension are reported only for compute nodes.
ClusterIdentifier	Filters requested data that is specific to the cluster. Metrics that are specific to clusters include HealthStatus , MaintenanceMode , and DatabaseConnections . General metrics for this dimension (for example, ReadIOPS) that are also metrics of nodes represent an aggregate of the node metric data. Take care in interpreting these metrics because they aggregate behavior of leader and compute nodes.
service class	The identifier for a WLM service class.
stage	The execution stages for a query. The possible values are as follows: • QueryPlanning: Time spent parsing and optimizing SQL statements. • QueryWaiting: Time spent waiting in the WLM queue. • QueryExecutingRead: Time spent executing read queries.

Dimension	Description
	• QueryExecutingInsert: Time spent executing insert queries. • QueryExecutingDelete: Time spent executing delete queries. • QueryExecutingUpdate: Time spent executing update queries. • QueryExecutingCtas: Time spent executing create table as queries. • QueryExecutingUnload: Time spent executing unload queries. • QueryExecutingCopy: Time spent executing copy queries. • QueryCommit: Time spent committing.
`wlmid`	The identifier for a workload management queue.
`QueryPriority`	The priority of the query. Possible values are `CRITICAL`, `HIGHEST`, `HIGH`, `NORMAL`, `LOW`, and `LOWEST`.
`QueueName`	The name of the workload management queue.
`FeatureType`	The feature that is limited by a usage limit. Possible values are `CONCURRENCY_SCALING` , `CROSS_REGION_DATASHARING` , and `SPECTRUM`.
`UsageLimitId`	The identifier for a usage limit.

Amazon Redshift query and load performance data

In addition to the CloudWatch metrics, Amazon Redshift provides query and load performance data. Query and load performance data can be used to help you understand the relation between database performance and cluster metrics. For example, if you notice that a cluster's CPU spiked, you can find the spike on the cluster CPU graph and see the queries that were running at that time. Conversely, if you are reviewing a specific query, metric data (like CPU) is displayed in context so that you can understand the query's impact on cluster metrics.

Query and load performance data are not published as CloudWatch metrics and can only be viewed in the Amazon Redshift console. Query and load performance data are generated from

querying with your database's system tables (for more information, see System tables reference in the *Amazon Redshift Developer Guide*). You can also generate your own custom database performance queries, but we recommend starting with the query and load performance data presented in the console. For more information about measuring and monitoring your database performance yourself, see Managing performance in the *Amazon Redshift Developer Guide.*

The following table describes different aspects of query and load data you can access in the Amazon Redshift console.

Query/Load data	Description
Query summary	A list of queries in a specified time period. The list can be sorted on values such as query ID, query runtime, and status. View this data in the **Query monitoring** tab of the cluster detail page.
Query detail	Provides details on a particular query including: • Query properties such as the query ID, type, cluster the query was run on, and runtime. • Details such as the status of the query and the number of errors. • The SQL statement that was run. • An explain plan if available. • Cluster performance data during the query execution (for more information, see Viewing query history data).
Load summary	Lists all the loads in a specified time period. The list can be sorted on values such as query ID, query runtime, and status. View this data in the **Query monitoring** tab of the cluster detail page.
Load detail	Provides details on a particular load operation including: • Load properties such as the query ID, type, cluster the query was run on, and runtime. • Details such as the status of the load and the number of errors. • The SQL statement that was run. • A list of loaded files.

Query/Load data	Description
	• Cluster performance data during the load operation (for more information, see Viewing query history data).

Viewing performance data

In this section, you can find how to view performance data in the Amazon Redshift console, which includes information about cluster and query performance. Additionally, you can create alarms on cluster metrics directly from the Amazon Redshift console.

When you view performance data in the Amazon Redshift console, you view it by cluster. The performance data graphs for a cluster are designed to give you access to data to answer your most common performance questions. For some performance data (see Performance data in Amazon Redshift), you can also use CloudWatch to further customize your metrics graphs. For example, you can choose longer times or combine metrics across clusters. For more information about working with the CloudWatch console, see Performance metrics in the CloudWatch console.

Watch the following video to learn how to monitor, isolate, and optimize your queries using the query monitoring features on the Amazon Redshift console: Query Monitoring with Amazon Redshift.

Topics

- Viewing cluster performance data
- Viewing query history data
- Viewing database performance data
- Viewing workload concurrency and concurrency scaling data
- Viewing automatic optimization data
- Viewing queries and loads
- Viewing and analyzing query details
- Viewing cluster performance as queries run
- Viewing cluster metrics during load operations
- Viewing the cluster workload breakdown chart

Viewing cluster performance data

By using cluster metrics in Amazon Redshift, you can do the following common performance tasks:

- Determine if cluster metrics are abnormal over a specified time range and, if so, identify the queries responsible for the performance hit.
- Check if historical or current queries are impacting cluster performance. If you identify a problematic query, you can view details about it including the cluster performance during the query's execution. You can use this information in diagnosing why the query was slow and what can be done to improve its performance.

To view performance data

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the name of a cluster from the list to open its details. The details of the cluster are displayed, which can include **Cluster performance**, **Query monitoring**, **Databases**, **Datashares**, **Schedules**, **Maintenance**, and **Properties** tabs.
3. Choose the **Cluster performance** tab for performance information including the following:

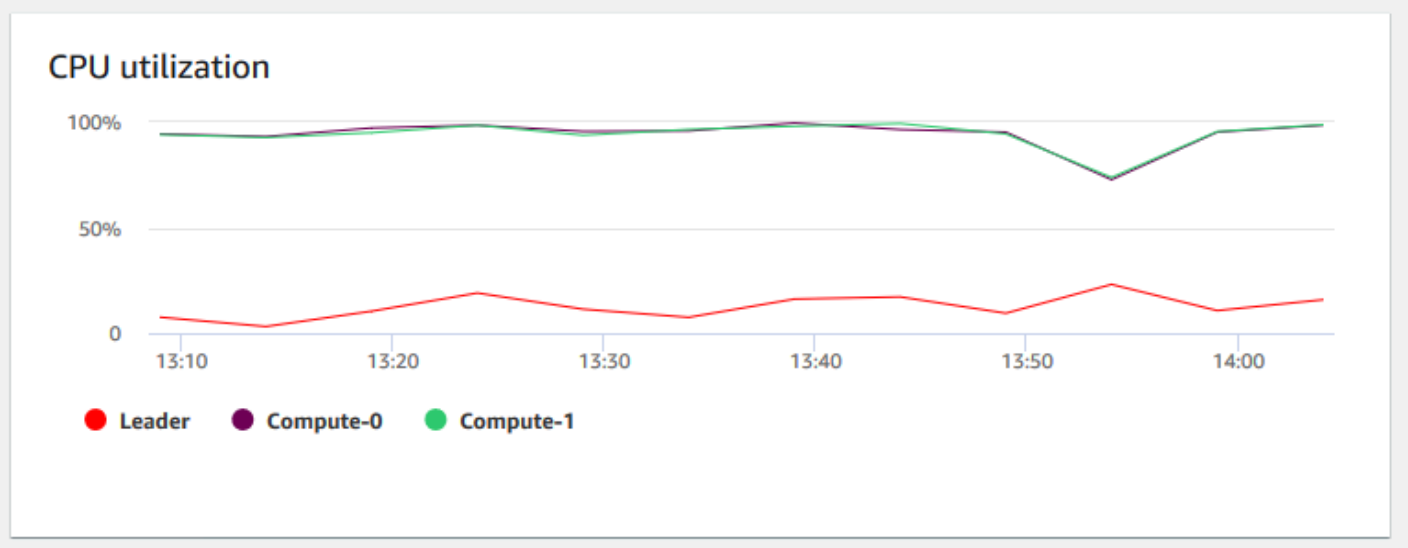
   - **CPU utilization**
   - **Percentage disk space used**
   - **Database connections**
   - **Health status**
   - **Query duration**
   - **Query throughput**
   - **Concurrency scaling activity**

   Many more metrics are available. To see the available metrics and choose which are displayed, choose the **Preferences** icon.

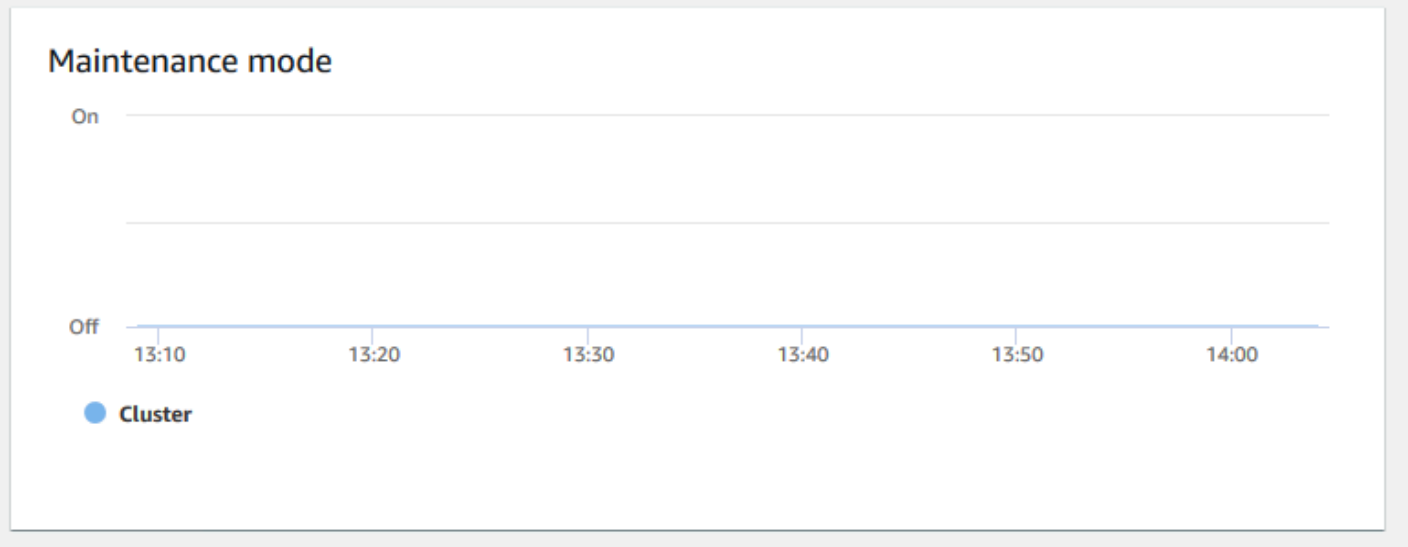
Cluster performance graphs

The following examples show some of the graphs that are displayed in the new Amazon Redshift console.

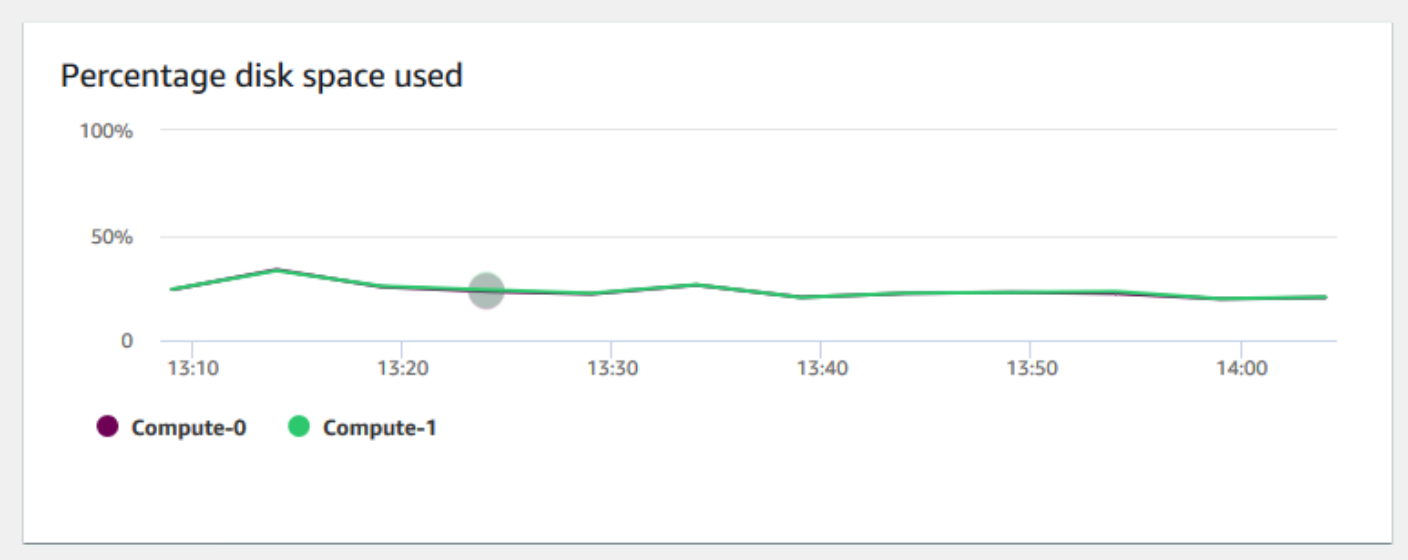
- **CPU utilization** – Shows the percentage of CPU utilization for all nodes (leader and compute). To find a time when the cluster usage is lowest before scheduling cluster migration or other resource-consuming operations, monitor this chart to see CPU utilization per individual or all of nodes.



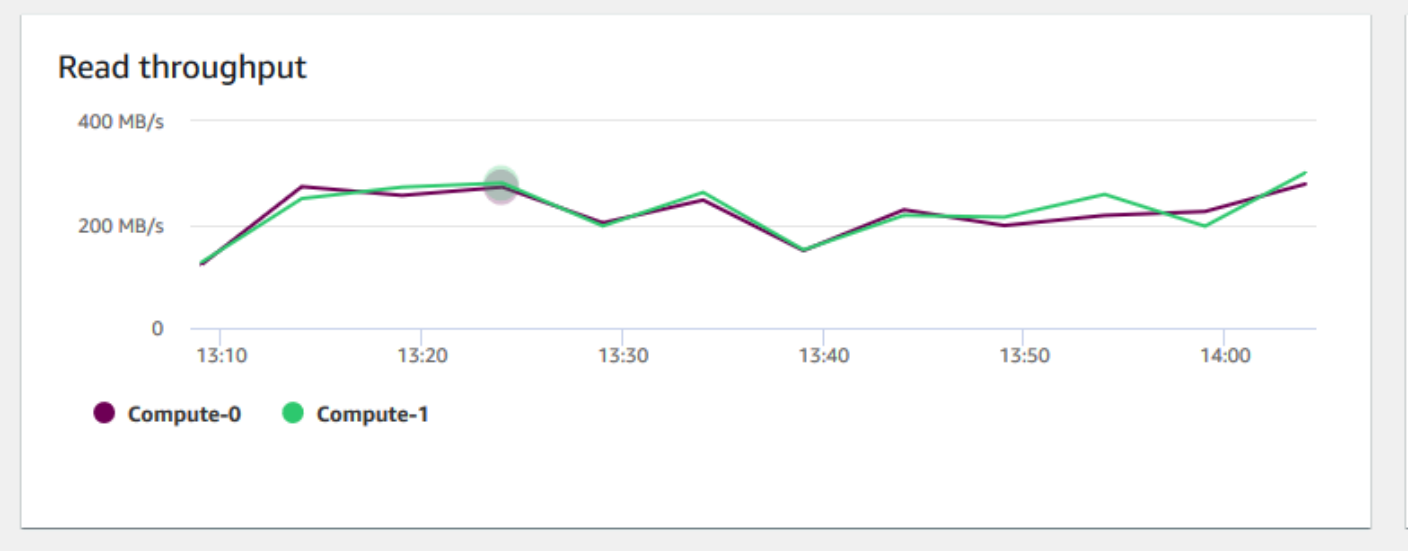
- **Maintenance mode** – Shows whether the cluster is in the maintenance mode at a chosen time by using `On` and `Off` indicators. You can see the time when the cluster is undergoing maintenance. You can then correlate this time to operations that are done to the cluster to estimate its future downtimes for recurring events.



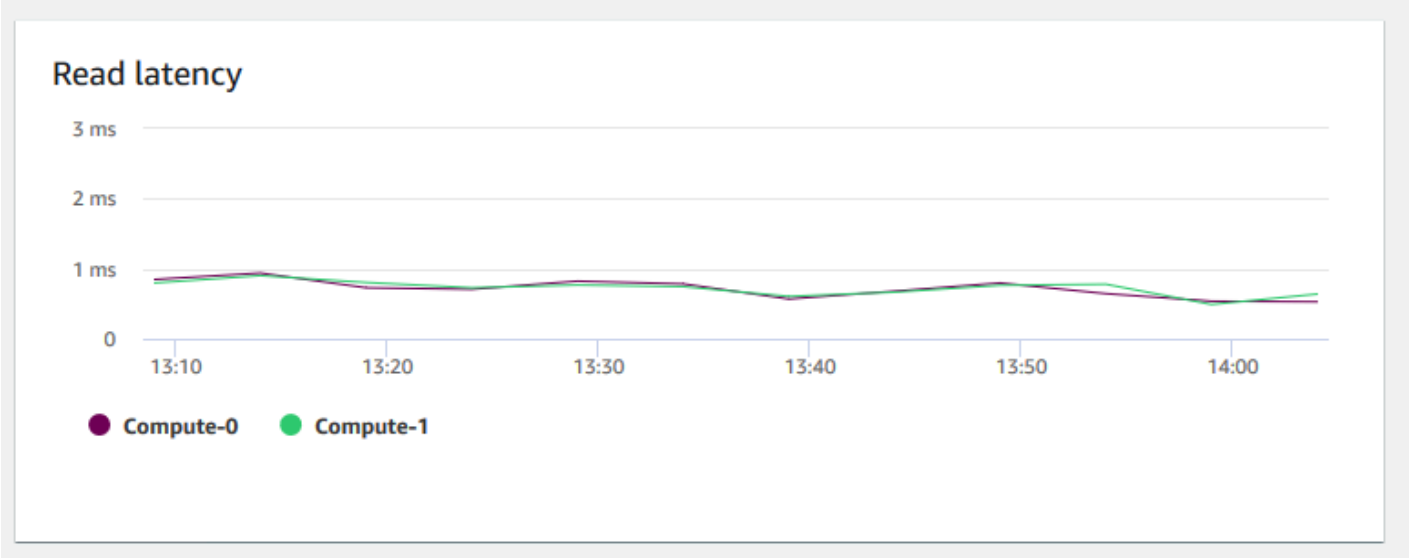
- **Percentage disk space used** – Shows the percentage of disk space usage per each compute node, and not for the cluster as a whole. You can explore this chart to monitor the disk utilization. Maintenance operations like VACUUM and COPY use intermediate temporary storage space for their sort operations, so a spike in disk usage is expected.



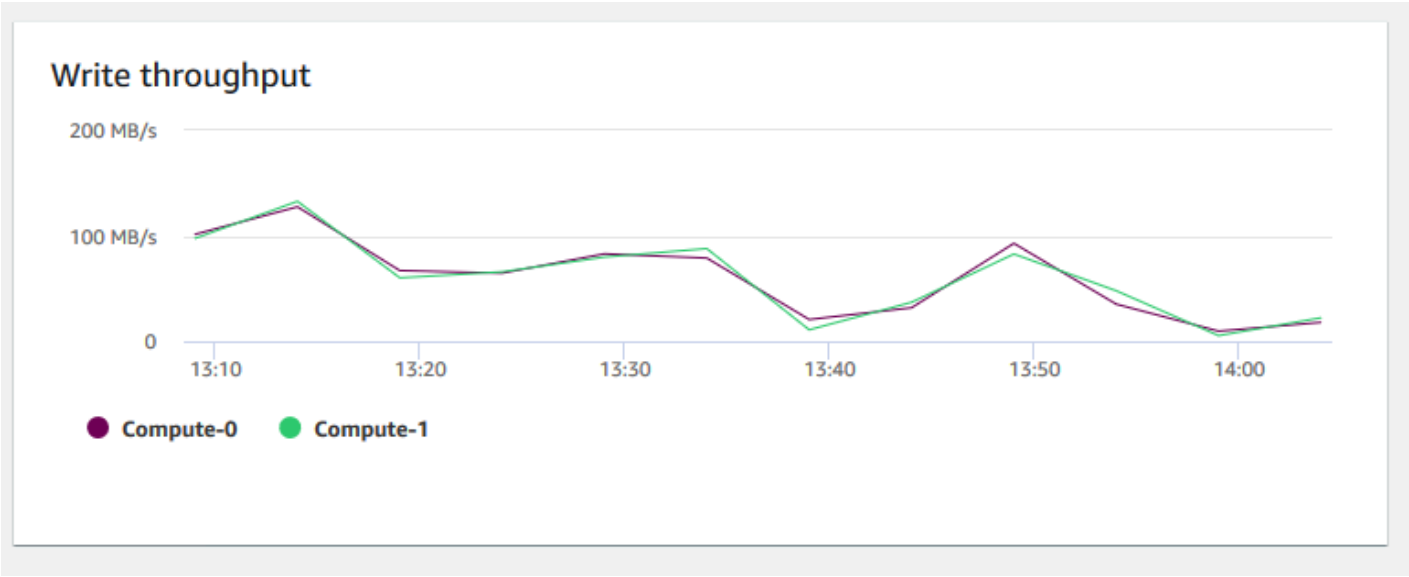
- **Read throughput** – Shows the average number of megabytes read from disk per second. You can evaluate this chart to monitor the corresponding physical aspect of the cluster. This throughput doesn't include network traffic between instances in the cluster and its volume.



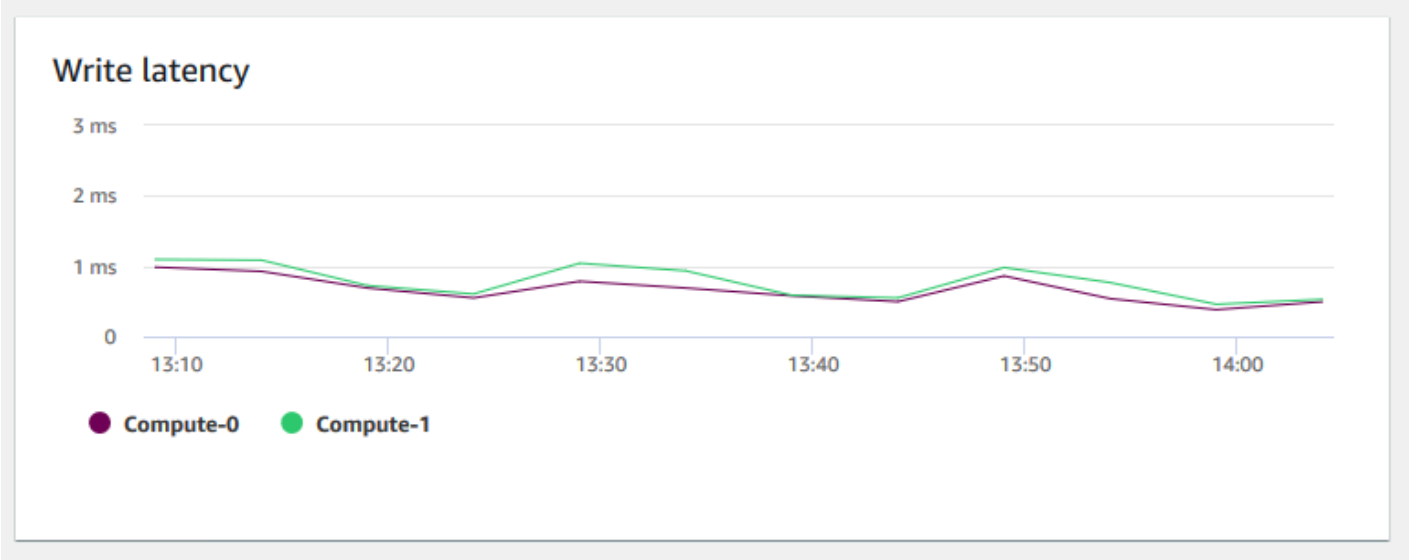
- **Read latency** – Shows the average amount of time taken for disk read I/O operations per millisecond. You can view the response times for the data to return. When latency is high, it means that the sender spends more time idle (not sending any new packets), which reduces how fast throughput grows.



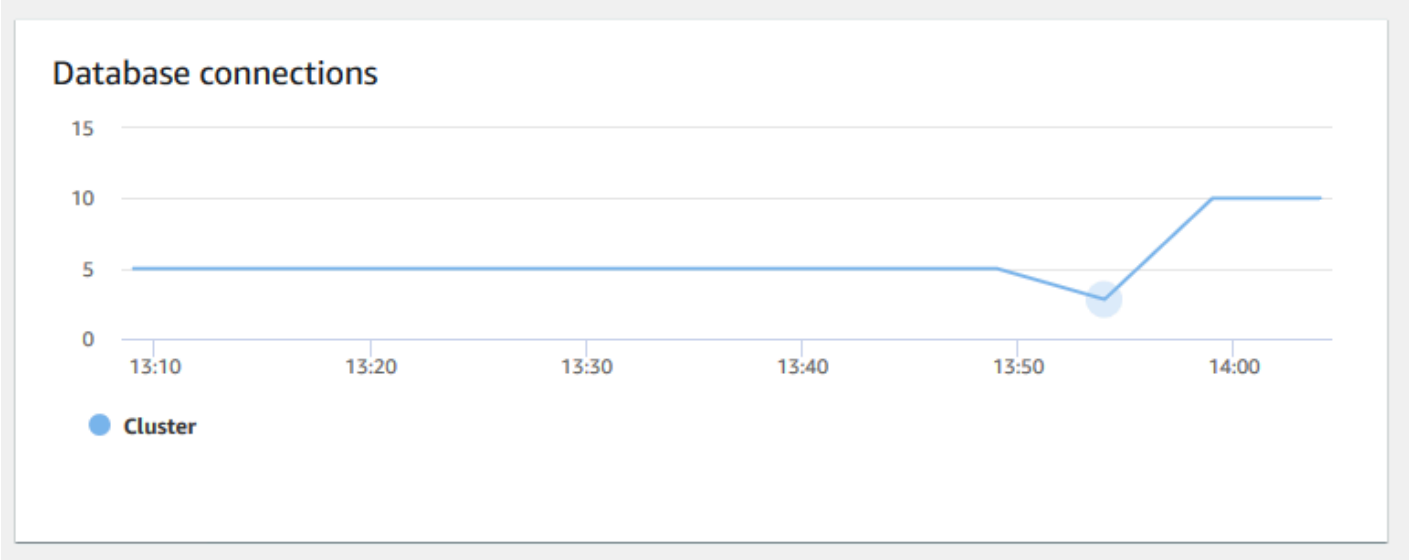
- **Write throughput** – Shows the average number of megabytes written to disk per second. You can evaluate this metric to monitor the corresponding physical aspect of the cluster. This throughput doesn't include network traffic between instances in the cluster and its volume.



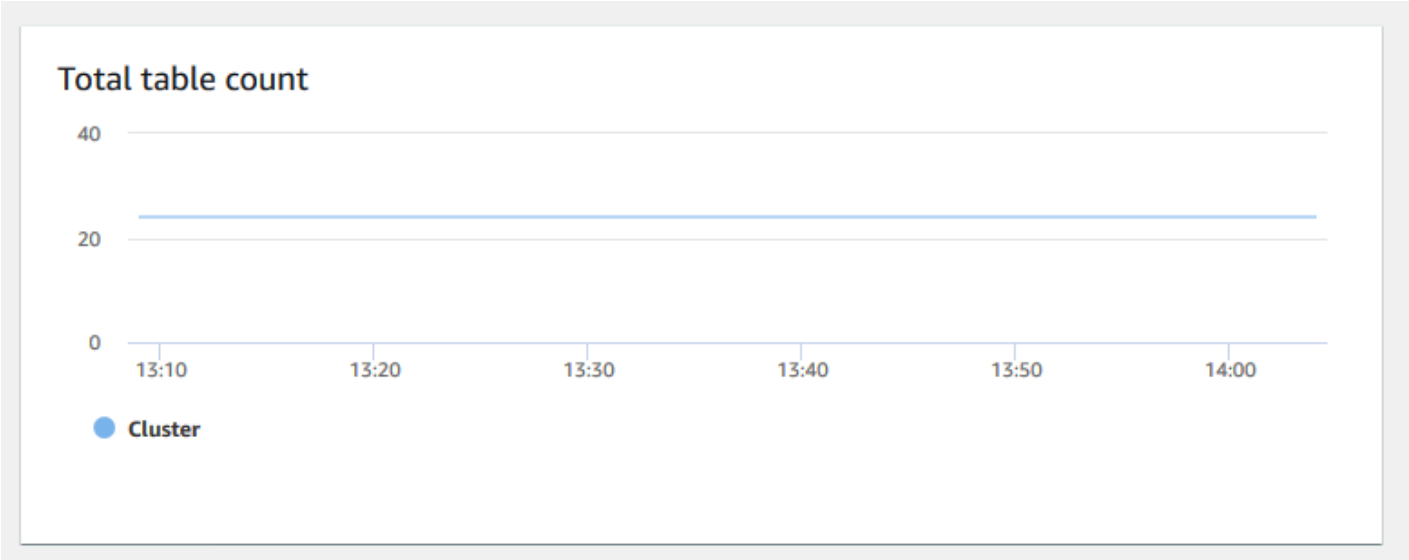
- **Write latency** – Shows the average amount of time in milliseconds taken for disk write I/O operations. You can evaluate the time for the write acknowledgment to return. When latency is high, it means that the sender spends more time idle (not sending any new packets), which reduces how fast throughput grows.



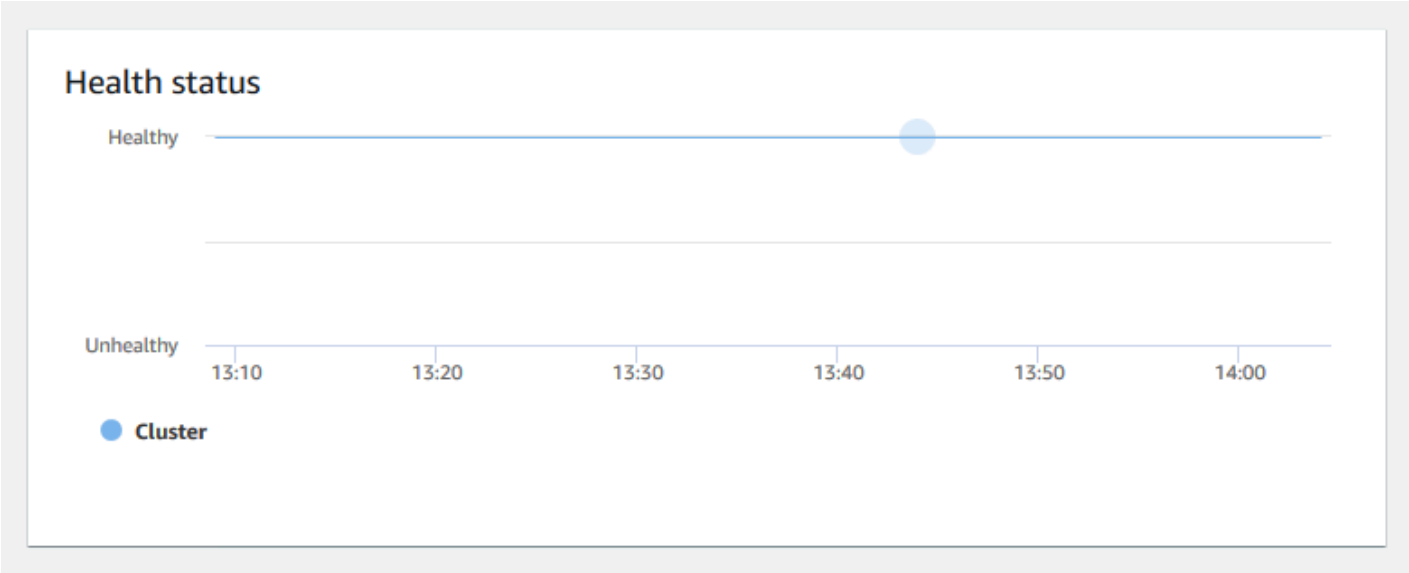
- **Database connections** – Shows the number of database connections to a cluster. You can use this chart to see how many connections are established to the database and find a time when the cluster usage is lowest.



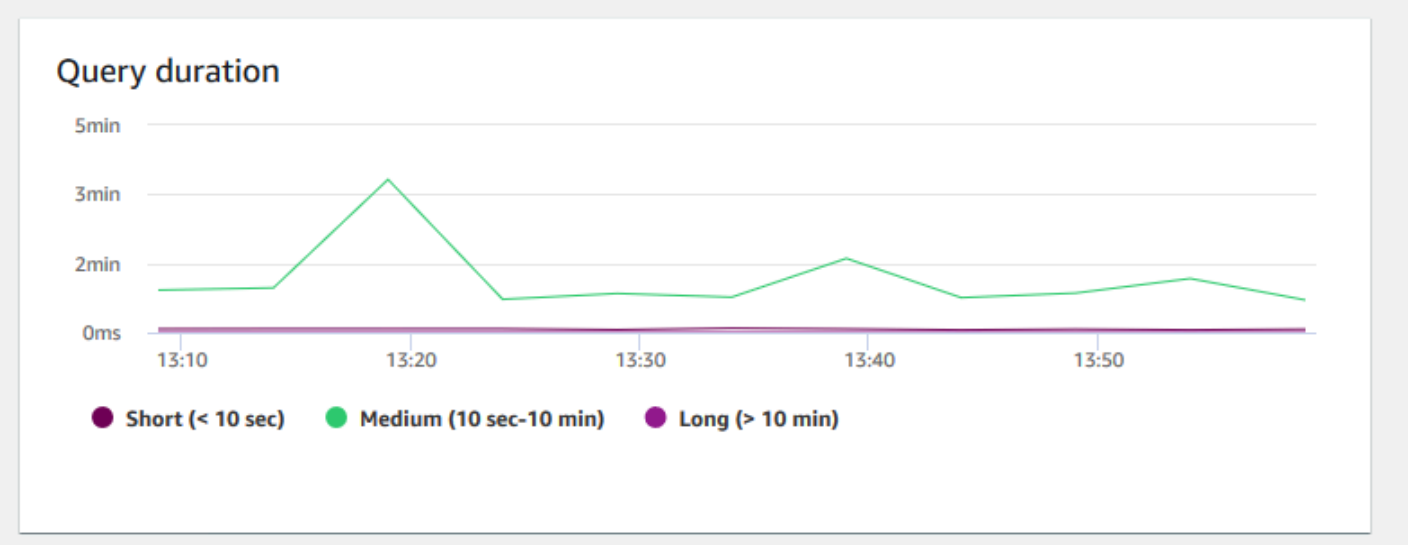
- **Total table count** – Shows the number of user tables open at a particular point in time within a cluster. You can monitor the cluster performance when open table count is high.



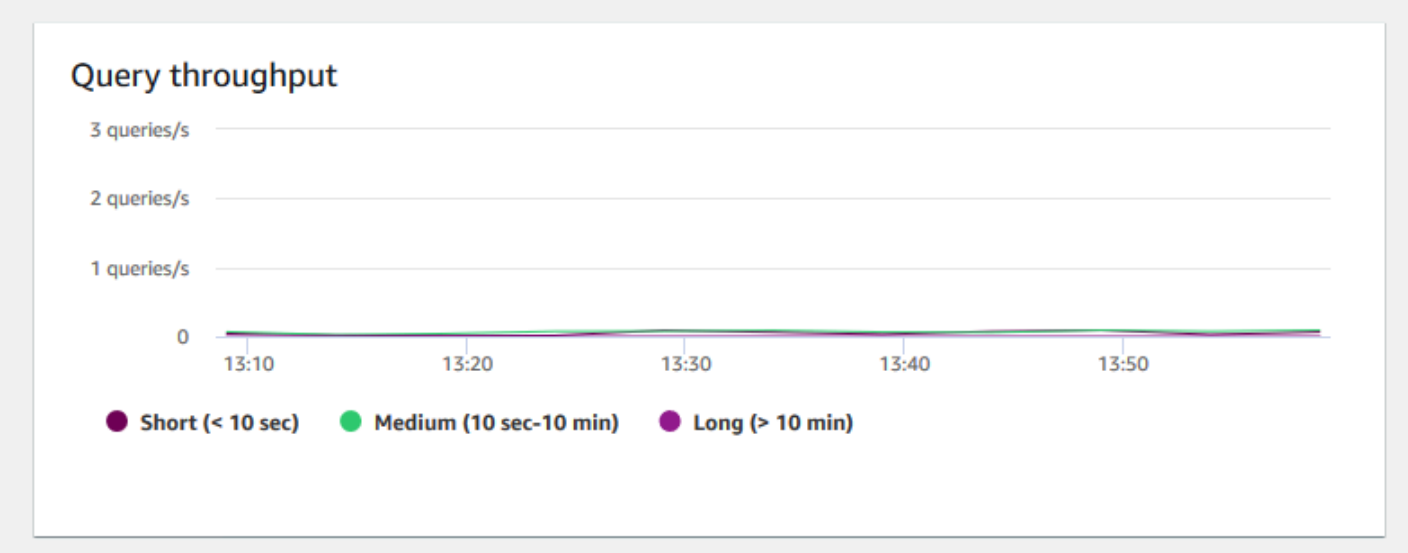
- **Health status** – Indicates the health of the cluster as `Healthy` or `Unhealthy`. If the cluster can connect to its database and performs a simple query successfully, the cluster is considered healthy. Otherwise, the cluster is unhealthy. An unhealthy status can occur when the cluster database is under extremely heavy load or if there is a configuration problem with a database on the cluster.



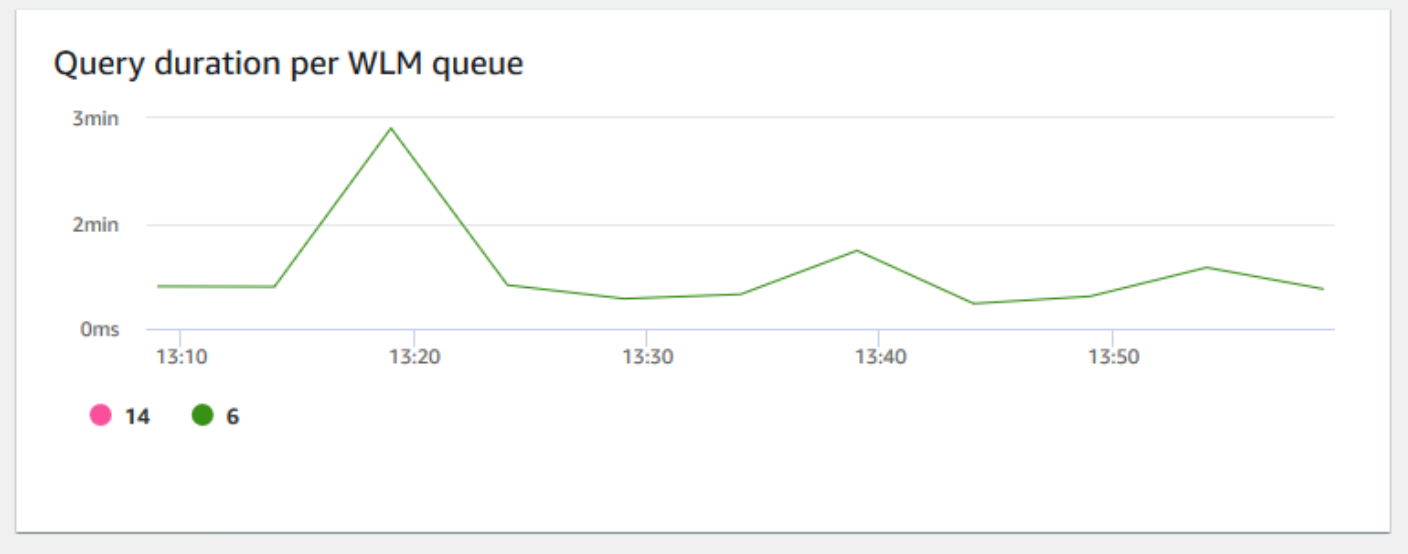
- **Query duration** – Shows the average amount of time to complete a query in microseconds. You can benchmark the data on this chart to measure I/O performance within the cluster and tune its most time-consuming queries if necessary.



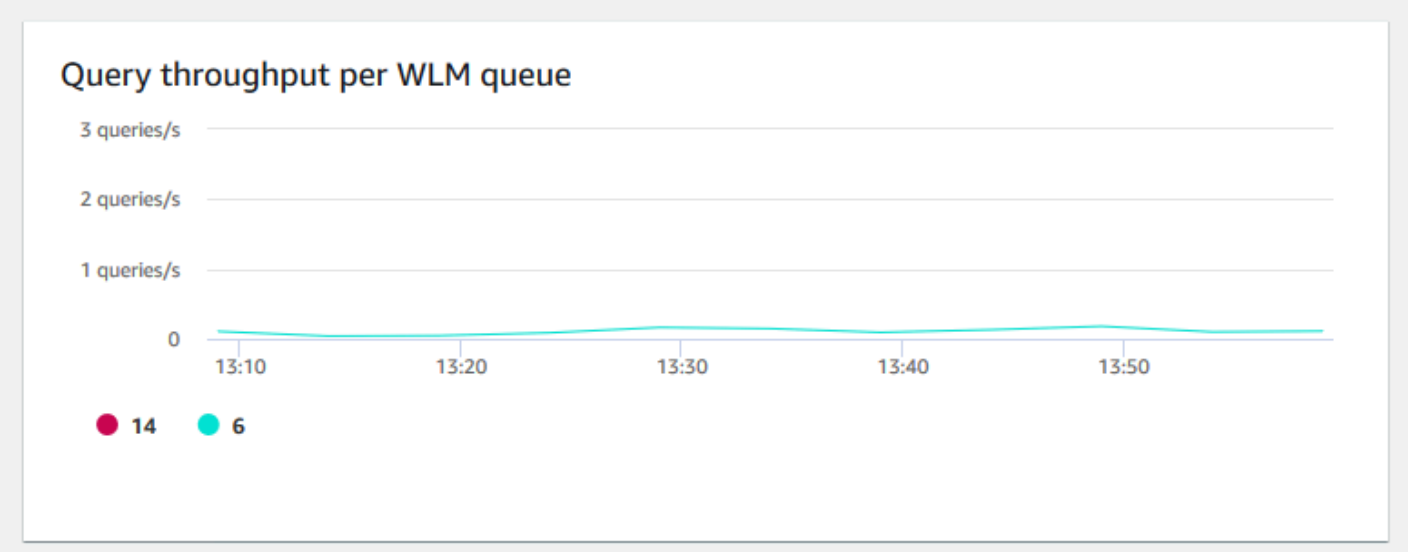
- **Query throughput** – Shows the average number of completed queries per second. You can analyze data on this chart to measure database performance and characterize the ability of the system to support a multiuser workload in a balanced way.



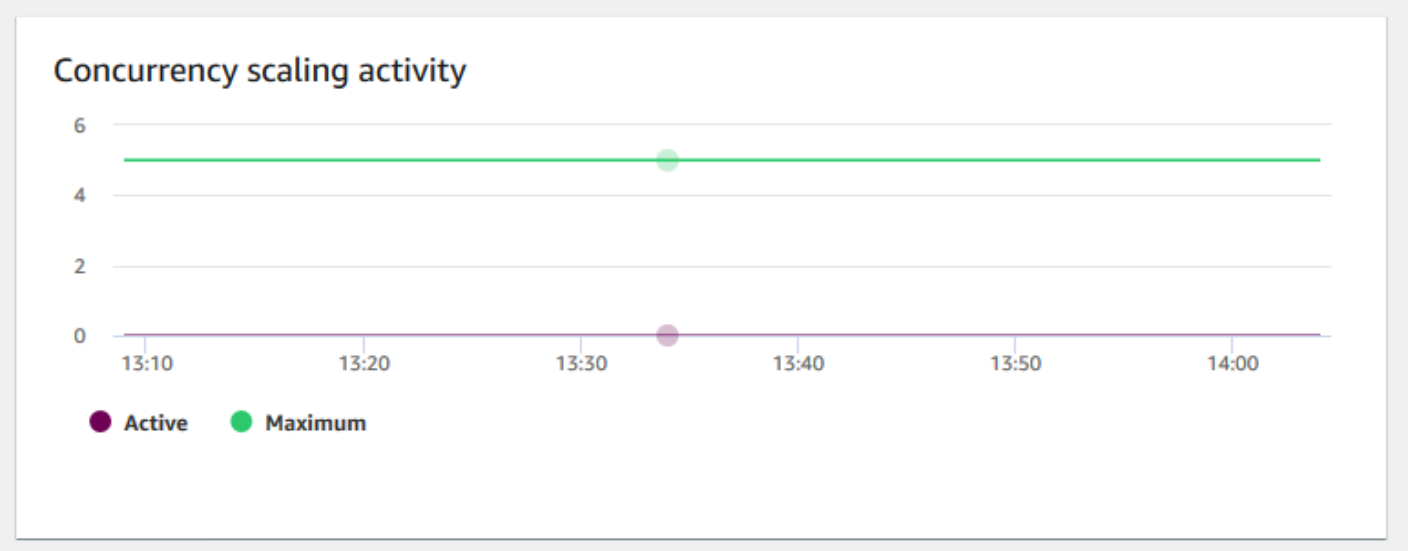
- **Query duration per WLM queue** – Shows the average amount of time to complete a query in microseconds. You can benchmark the data on this chart to measure I/O performance per WLM queue and tune its most time-consuming queries if necessary.



- **Query throughput per WLM queue** – Shows the average number of completed queries per second. You can analyze data on this chart to measure database performance per WLM queue.



- **Concurrency scaling activity** – Shows the number of active concurrency scaling clusters. When concurrency scaling is enabled, Amazon Redshift automatically adds additional cluster capacity when you need it to process an increase in concurrent read queries.



Viewing query history data

You can use query history metrics in Amazon Redshift to do the following:

- Isolate and diagnose query performance problems.
- Compare query runtime metrics and cluster performance metrics on the same timeline to see how the two might be related. Doing so helps identify poorly performing queries, look for bottleneck queries, and determine if you need to resize your cluster for your workload.
- Drill down to the details of a specific query by choosing it in the timeline. When **Query ID** and other properties are displayed in a row below the graph, then you can choose the query to see query details. Details include, for example, the query's SQL statement, execution details, and query plan. For more information, see Viewing and analyzing query details.
- Determine if your load jobs complete successfully and meet your service level agreements (SLAs).

To display query history data

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the cluster name from the list to open its details. The details of the cluster are displayed, which can include **Cluster performance**, **Query monitoring**, **Databases**, **Datashares**, **Schedules**, **Maintenance**, and **Properties** tabs.
3. Choose the **Query monitoring** tab for metrics about your queries.
4. In the **Query monitoring** section, choose the **Query history** tab.

Using controls on the window, you can toggle between **Query list** and **Cluster metrics**.

When you choose **Query list**, the tab includes the following graphs:

- **Query runtime** – The query activity on a timeline. Use this graph to see which queries are running in the same timeframe. Choose a query to view more query execution details. The x-axis shows the selected period. You can filter the graphed queries by running, completed, loads, and so on. Each bar represents a query, and the length of the bar represents its runtime from the start of the bar to the end. The queries can include SQL data manipulation statements (such as SELECT, INSERT, DELETE) and loads (such as COPY). By default, the top 100 longest running queries are shown for the selected time period.
- **Queries and loads** – List of queries and loads that ran on the cluster. The window includes an option to **Terminate query** if a query is currently running.

When you choose **Cluster metrics**, the tab includes the following graphs:

- **Query runtime** – The query activity on a timeline. Use this graph to see which queries are running in the same timeframe. Choose a query to view more query execution details.
- **CPU utilization** – The CPU utilization of the cluster by leader node and average of compute nodes.
- **Storage capacity used** – The percent of the storage capacity used.
- **Active database connections** – The number of active database connections to the cluster.

Consider the following when working with the query history graphs:

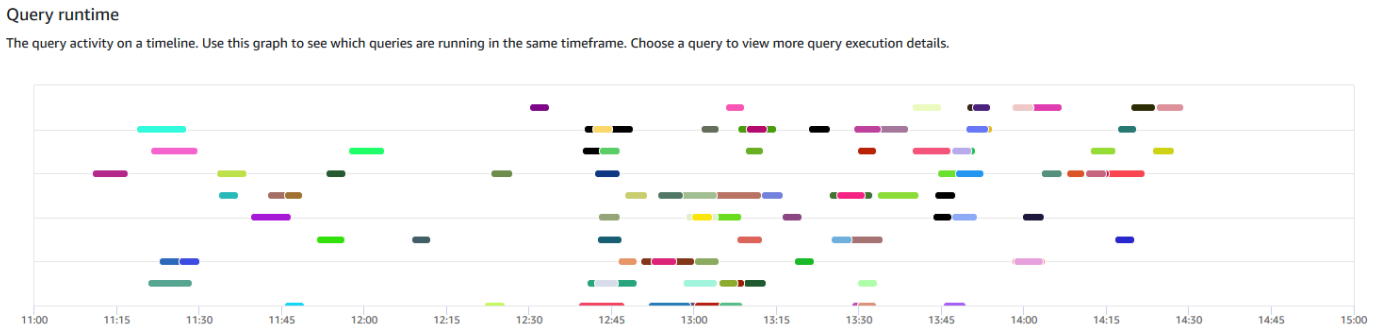
- Choose a bar that represents a specific query on the **Query runtime** chart to see details about that query. You can also, choose a query ID on **Queries and loads** list to see its details.
- You can swipe to select a section of the **Query runtime** chart to zoom in to display a specific time period.
- On the **Query runtime** chart, to have all data considered by your chosen filter, page forward through all pages listed on the **Queries and loads** list.
- You can change which columns and the number of rows displayed on the **Queries and loads** list using the preferences window displayed by the **settings gear icon**.
- The **Queries and loads** list can also be displayed by navigating from the left navigator **Queries** icon, **Queries and loads**. For more information, see Viewing queries and loads.

Query history graphs

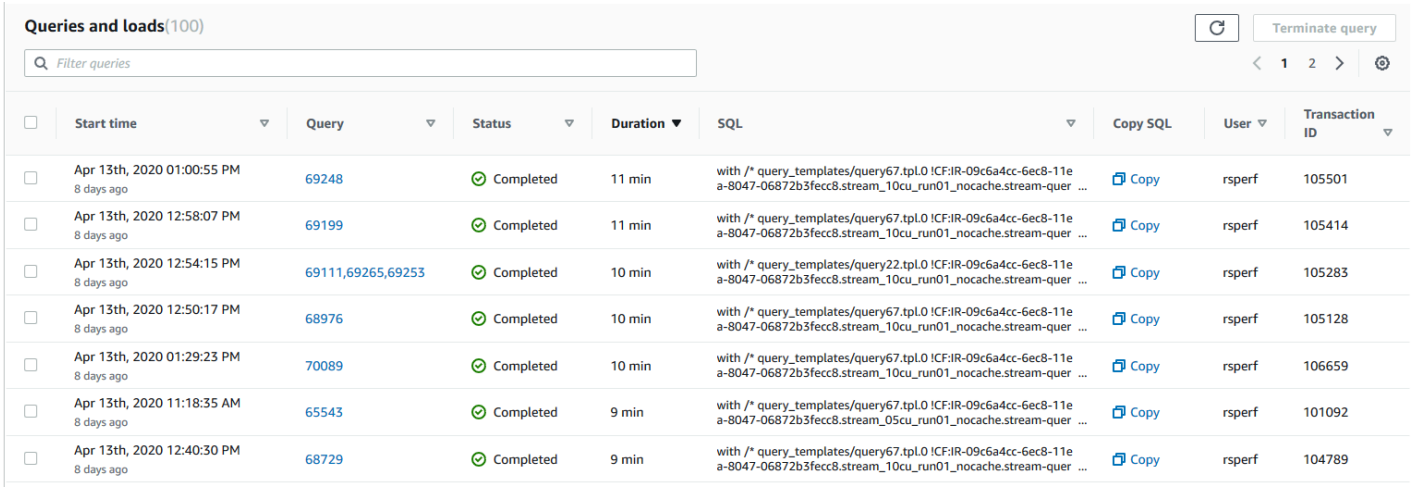
The following examples show graphs that are displayed in the new Amazon Redshift console.

> **Note**
>
> The Amazon Redshift console graphs only contain data for the latest 100,000 queries.

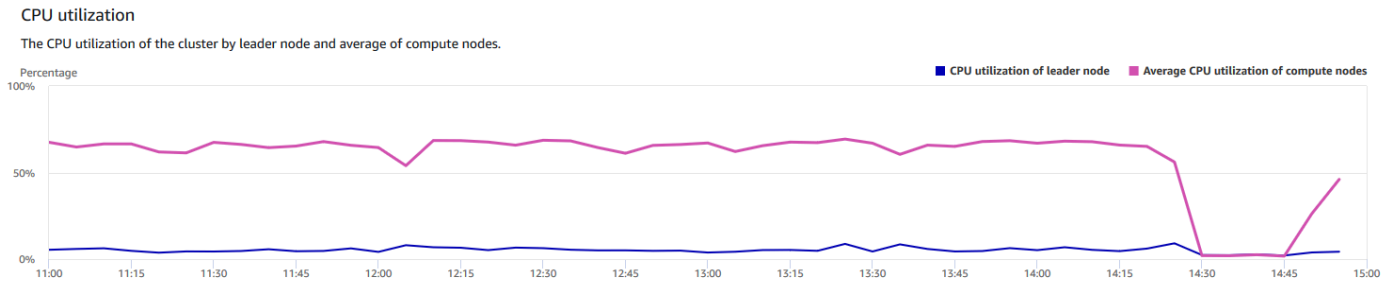
- **Query runtime**



- **Queries and loads**



- **CPU utilization**



- **Storage capacity used**



- **Active database connections**



Viewing database performance data

You can use database performance metrics in Amazon Redshift to do the following:

- Analyze the time spent by queries by processing stages. You can look for unusual trends in the amount of time spent in a stage.
- Analyze the number of queries, duration, and throughput of queries by duration ranges (short, medium, long).
- Look for trends in the about of query wait time by query priority (Lowest, Low, Normal, High, Highest, Critical).
- Look for trends in the query duration, throughput, or wait time by WLM queue.

To display database performance data

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the cluster name from the list to open its details. The details of the cluster are displayed, including **Cluster performance**, **Query monitoring**, **Databases**, **Datashares**, **Schedules**, **Maintenance**, and **Properties** tabs.

3. Choose the **Query monitoring** tab for metrics about your queries.
4. In the **Query monitoring** section, choose **Database performance** tab.

   Using controls on the window, you can toggle between **Cluster metrics** and **WLM queue metrics**.

   When you choose **Cluster metrics**, the tab includes the following graphs:

   - **Workload execution breakdown** – The time used in query processing stages.
   - **Queries by duration range** – The number of short, medium, and long queries.
   - **Query throughput** – The average number of queries completed per second.
   - **Query duration** – The average amount of time to complete a query.
   - **Average queue wait time by priority** – The total time queries spent waiting in the WLM queue by query priority.

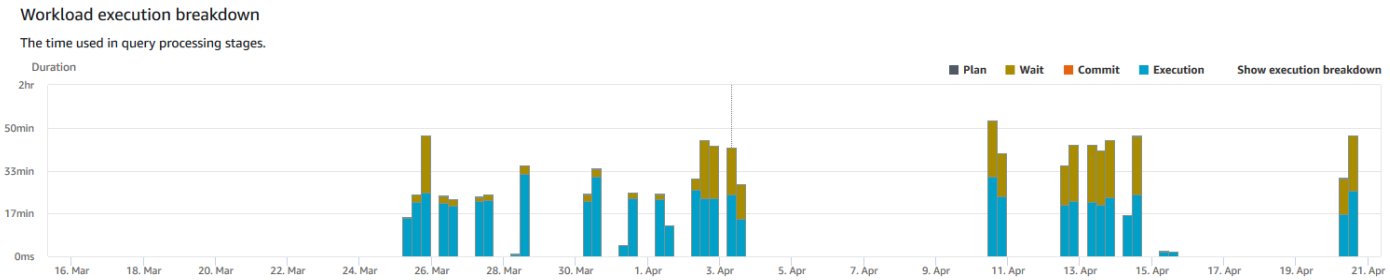
   When you choose **WLM queue metrics**, the tab includes the following graphs:

   - **Query duration by queue** – The average query duration by WLM queue.
   - **Query throughput by queue** – The average number of queries completed per second by WLM queue.
   - **Query wait time by queue** – The average duration of queries spent waiting by WLM queue.

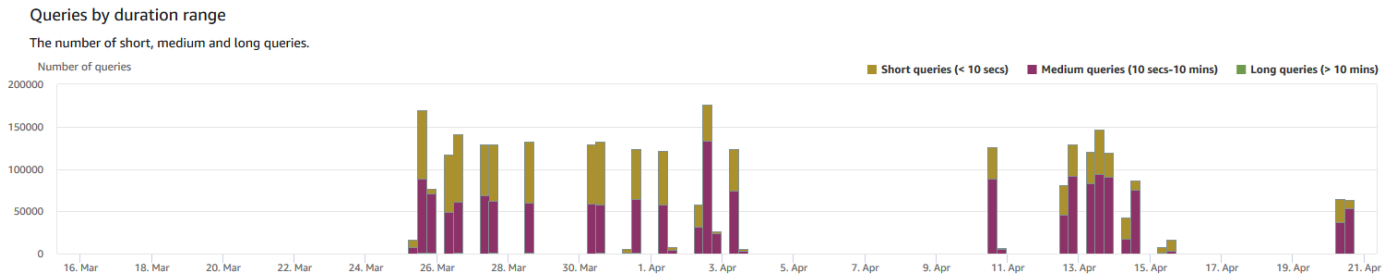
Database performance graphs

The following examples show graphs that are displayed in the new Amazon Redshift console.

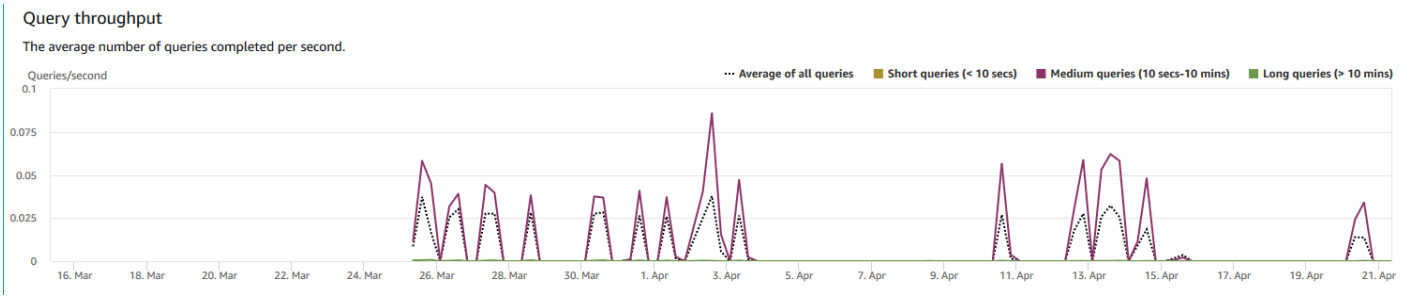
- **Workload execution breakdown**



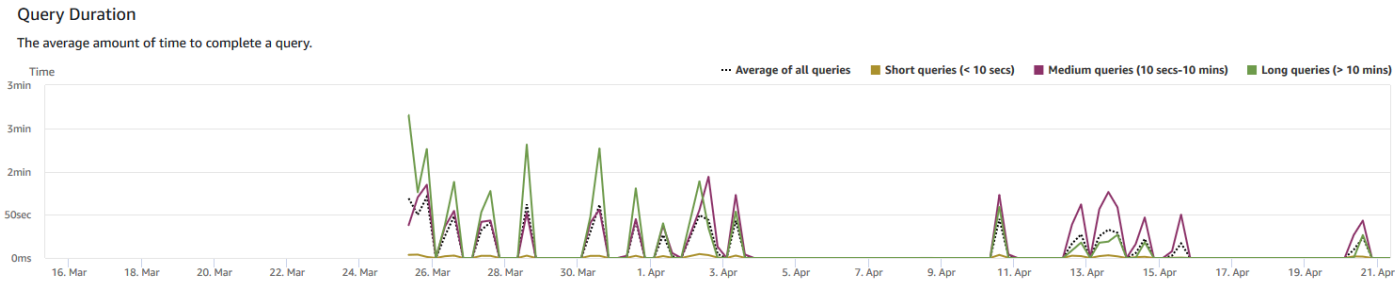
- **Queries by duration range**



- **Query throughput**



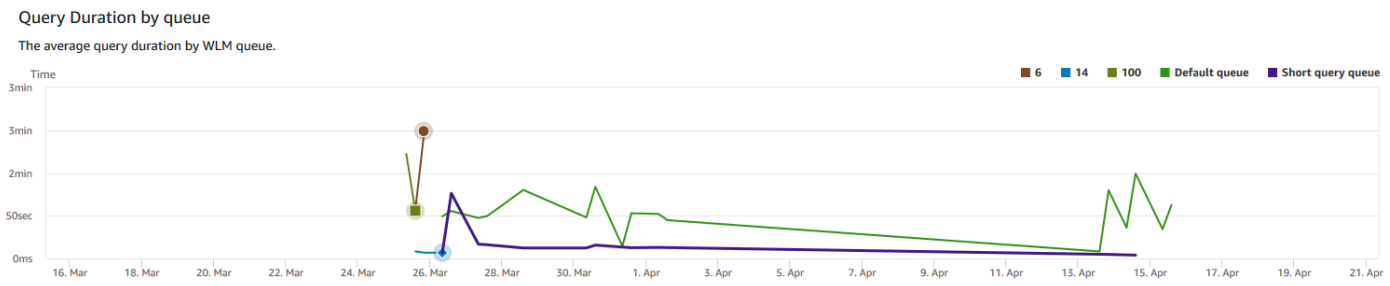
- **Query duration**



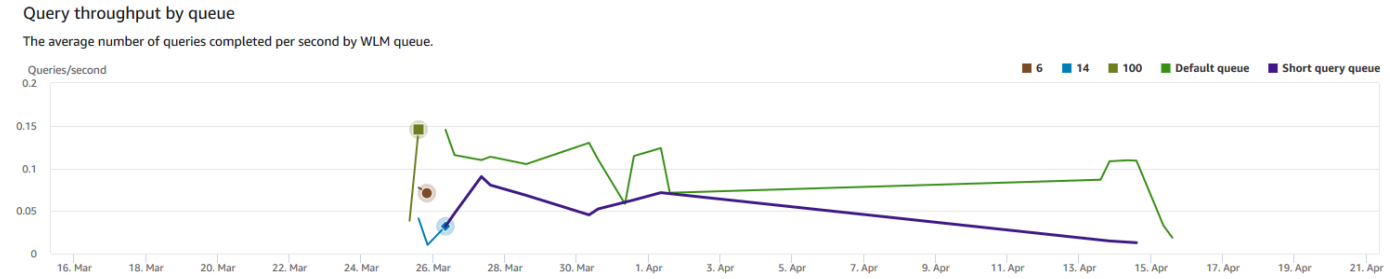
- **Average queue wait time by priority**



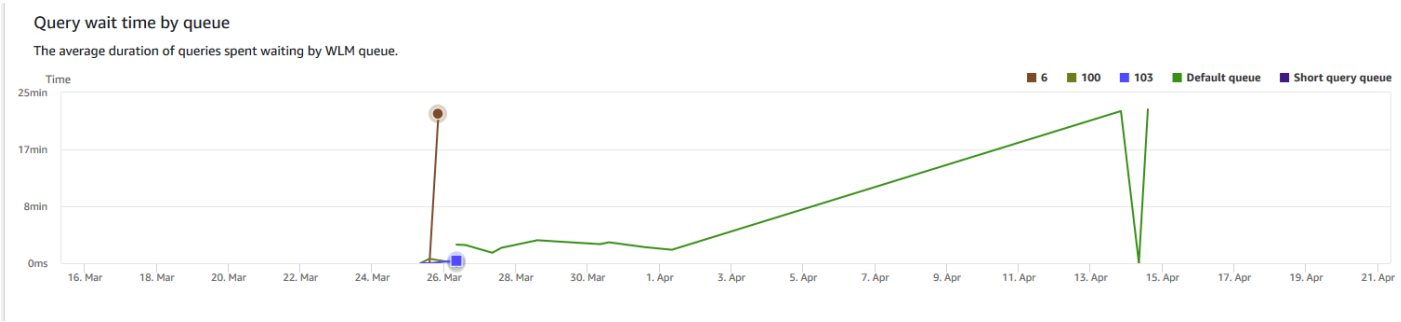
- **Query duration by queue**



- **Query throughput by queue**



- **Query wait time by queue**



Viewing workload concurrency and concurrency scaling data

By using concurrency scaling metrics in Amazon Redshift, you can do the following:

- Analyze whether you can reduce the number of queued queries by enabling concurrency scaling. You can compare by WLM queue or for all WLM queues.
- View concurrency scaling activity in concurrency scaling clusters. This can tell you if concurrency scaling is limited by the `max_concurrency_scaling_clusters`. If so, you can choose to increase the `max_concurrency_scaling_clusters` in the DB parameter.
- View the total usage of concurrency scaling summed across all concurrency scaling clusters.

To display concurrency scaling data

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the cluster name from the list to open its details. The details of the cluster are displayed, which can include **Cluster performance**, **Query monitoring**, **Databases**, **Datashares**, **Schedules**, **Maintenance**, and **Properties** tabs.
3. Choose the **Query monitoring** tab for metrics about your queries.
4. In the **Query monitoring** section, choose **Workload concurrency** tab.

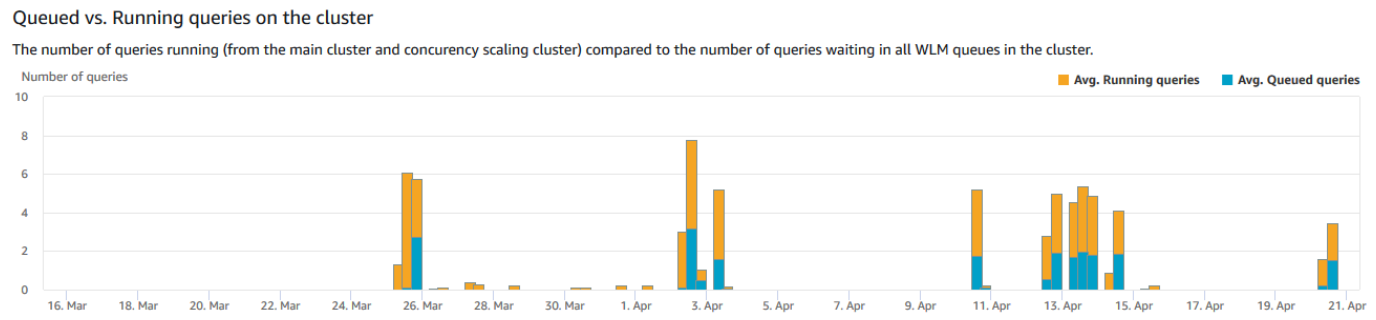
   The tab includes the following graphs:

   - **Queued vs. Running queries on the cluster** – The number of queries running (from the main cluster and concurrency scaling cluster) compared to the number of queries waiting in all WLM queues in the cluster.
   - **Queued vs. Running queries per queue** – The number of queries running (from the main cluster and concurrency scaling cluster) compared to the number or queries waiting in each WLM queue.
   - **Concurrency scaling activity** – The number of concurrency scaling clusters that are actively processing queries.
   - **Concurrency scaling usage** – The usage of concurrency scaling clusters that have active query processing activity.

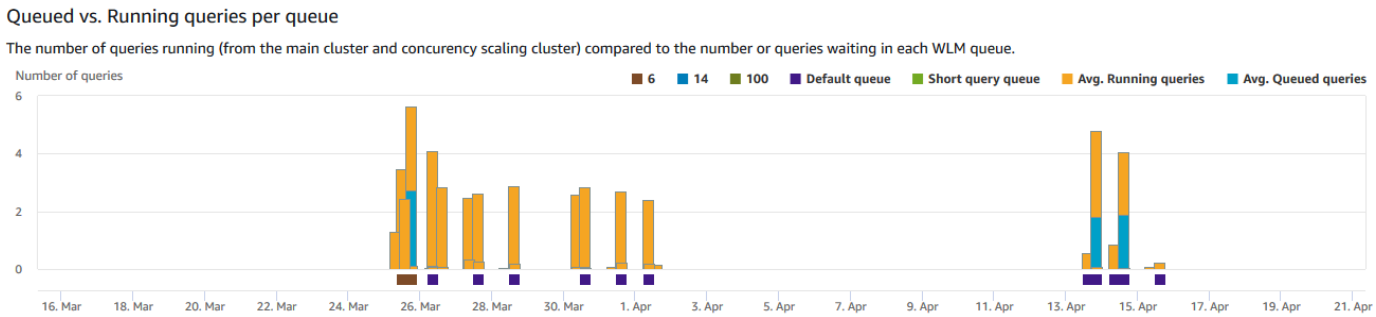
Workload concurrency graphs

The following examples show graphs that are displayed in the new Amazon Redshift console. To create similar graphs in Amazon CloudWatch, you can use the concurrency scaling and WLM CloudWatch metrics. For more information about CloudWatch metrics for Amazon Redshift, see Performance data in Amazon Redshift.

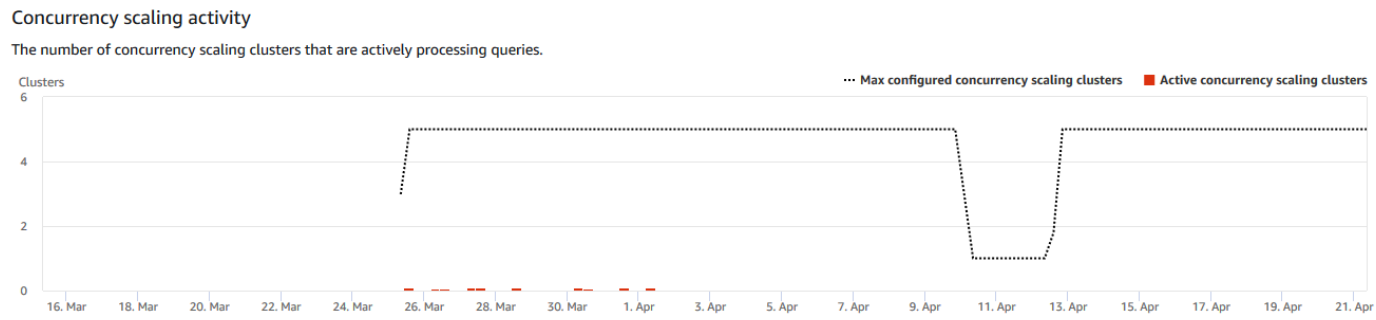
- **Queued vs. Running queries on the cluster**



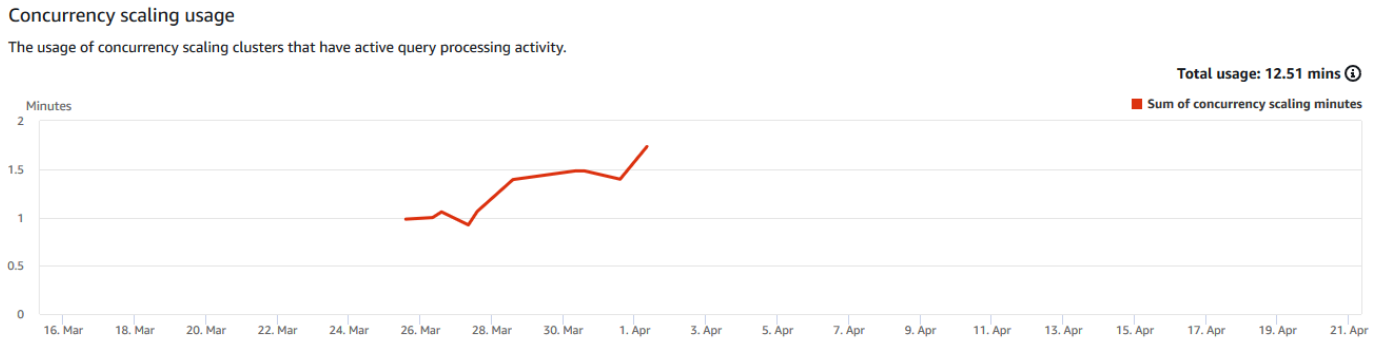
- **Queued vs. Running queries per queue**



- **Concurrency scaling activity**



- **Concurrency scaling usage**



Viewing automatic optimization data

The Amazon Redshift console provides information about automatic optimizations, or autonomics, run using extra compute resources. You can use this information to track usage and monitor whether usage limits have been reached. Though Amazon Redshift doesn't bill you for autonomics run on the provisioned cluster itself, it does bill you for autonomics run using extra compute resources. For more information, see Allocating extra compute resources for automatic database optimizations in the *Amazon Redshift Database Developer Guide*.

To view extra compute autonomics data

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the name of a cluster from the list to open its details.
3. From the cluster's details page, select **Manage usage limit** from the **Actions** drop-down menu. You can also select the **Maintenance** tab for a cluster, then scroll down and select **Create usage limits**.
4. The graph showing extra compute autonomics data appears under the section titled **Usage limit for extra compute for automatic optimization**. The graph displays the amount of time that Amazon Redshift runs autonomics using extra compute resources in a given time period.

Viewing queries and loads

The Amazon Redshift console provides information about queries and loads that run in the database. You can use this information to identify and troubleshoot queries that take a long time to process and that create bottlenecks preventing other queries from processing efficiently. You can use the queries information in the Amazon Redshift console to monitor query processing.

To display query performance data

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Queries and loads** to display the list of queries for your account.

   By default, the list displays queries for all your clusters over the past 24 hours. You can change the scope of the displayed date in the console.

> **Important**
>
> The **Queries and loads** list displays the longest running queries in the system, up to 100 queries.

Viewing and analyzing query details

With a query identifier, you can view details of a query. Details can include, for example, the query's completion status, duration, SQL statement and whether it's a user query or one that was rewritten by Amazon Redshift. A *user query* is a query that is submitted to Amazon Redshift, either from an SQL client or generated by a business intelligence tool. Amazon Redshift might rewrite the query to optimize it, and this can result in multiple rewritten queries. Although the process is done by Amazon Redshift, you see the rewritten queries on the query details page along with the user query.

To view a query

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Queries and loads** to display the list of queries for your account. You might need to change settings on this page to find your query.
3. Choose the **Query** identifier in the list to display **Query details**.

   The **Query details** page includes **Query details** and **Query plan** tabs with metrics about the query.

   Metrics include details about a query such as start time, query ID, status, and duration. Other details include whether a query ran on a main cluster or a concurrency scaling cluster, and if it's a parent or rewritten query.

Viewing cluster performance as queries run

You can monitor the performance of your clusters as queries run to identify potential bottlenecks and optimize query execution. Viewing cluster performance as queries run provides a real-time view of the system-level metrics, such as CPU utilization, disk I/O, and network traffic, as well as query-level details like execution time, data processed, and query steps. The following procedures

guides you through accessing and interpreting the performance metrics to effectively manage and optimize your provisioned clusters.

To display cluster performance as queries run

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the cluster name from the list to open its details. The details of the cluster are displayed, which can include **Cluster performance**, **Query monitoring**, **Databases**, **Datashares**, **Schedules**, **Maintenance**, and **Properties** tabs.
3. Choose the **Query monitoring** tab for more details.

   For more information, see Viewing query history data.

Viewing cluster metrics during load operations

When you view cluster performance during load operations, you can identify queries that are consuming resources and act to mitigate their effect. You can terminate a load if you don't want it to run to completion.

> **Note**
>
> The ability to terminate queries and loads in the Amazon Redshift console requires specific permission. If you want users to be able to terminate queries and loads, make sure to add the `redshift:CancelQuerySession` action to your AWS Identity and Access Management (IAM) policy. This requirement applies whether you select the **Amazon Redshift Read Only** AWS-managed policy or create a custom policy in IAM. Users who have the **Amazon Redshift Full Access** policy already have the necessary permission to terminate queries and loads. For more information about actions in IAM policies for Amazon Redshift, see Managing access to resources.

To display cluster performance during load operations

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.

2. On the navigation menu, choose **Clusters**, then choose the cluster name from the list to open its details. The details of the cluster are displayed, which can include **Cluster performance**, **Query monitoring**, **Databases**, **Datashares**, **Schedules**, **Maintenance**, and **Properties** tabs.
3. Choose the **Query monitoring** tab for more details.
4. In the **Queries and loads** section, choose **Loads** to view the load operations of a cluster. If a load is running, you can end it by choosing **Terminate query**.

Viewing the cluster workload breakdown chart

You can get a detailed view of your workload's performance by looking at the Workload execution breakdown chart in the console. We build the chart with data provided by the QueryRuntimeBreakdown metric. With this chart, you can see how much time your queries spend in the various processing stages, such as waiting and planning.

> **Note**
>
> The Workload execution breakdown chart isn't shown for single-node clusters.

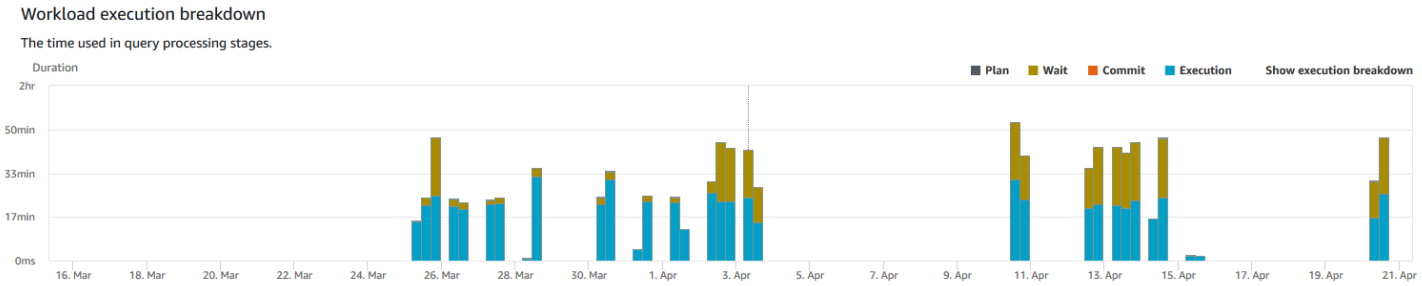
The following list of metrics describes the various processing stages:

- `QueryPlanning`: Time spent parsing and optimizing SQL statements.
- `QueryWaiting`: Time spent waiting in the workload management (WLM) queue.
- `QueryExecutingRead`: Time spent running read queries.
- `QueryExecutingInsert`: Time spent running insert queries.
- `QueryExecutingDelete`: Time spent running delete queries.
- `QueryExecutingUpdate`: Time spent running update queries.
- `QueryExecutingCtas`: Time spent running CREATE TABLE AS queries.
- `QueryExecutingUnload`: Time spent running unload queries.
- `QueryExecutingCopy`: Time spent running copy queries.

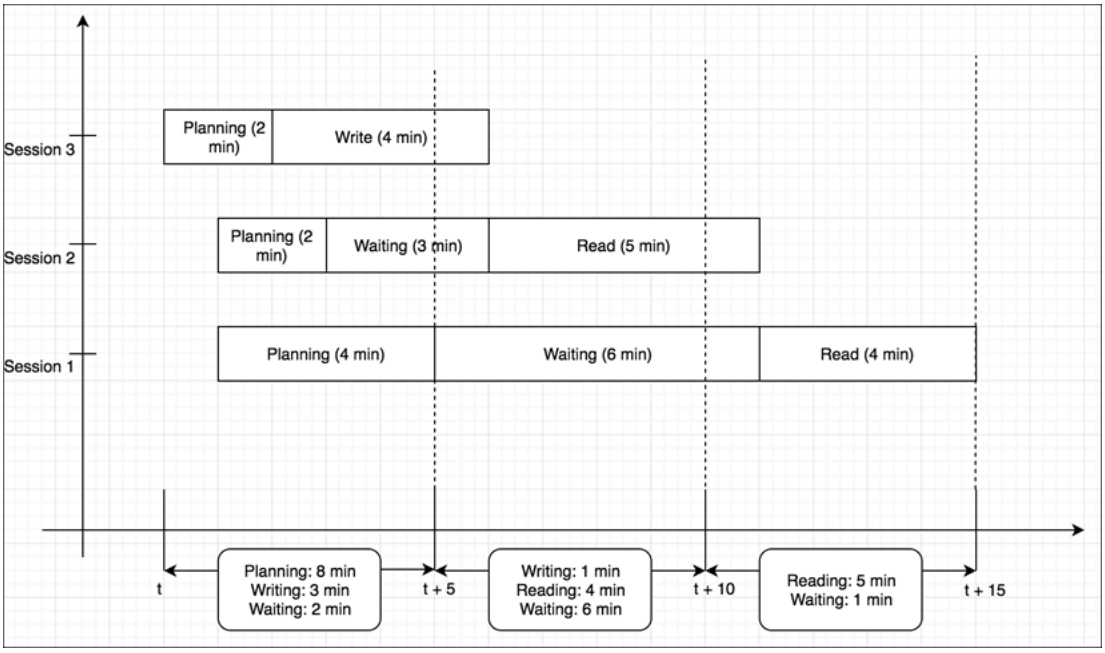
For example, the following graph in the Amazon Redshift console shows the amount of time that queries have spent in the plan, wait, read, and write stages. You can combine the findings from this graph with other metrics for further analysis. In some cases, your graph might show that queries with a short duration (as measured by the `QueryDuration` metric) are spending a long time in

the wait stage. In these cases, you can increase the WLM concurrency rate for a particular queue to increase throughput.

Following, is an example of the workload execution breakdown chart. In the chart, the y-axis value is the average duration of each stage at the specified time shown as a stacked bar graph.



The following diagram illustrates how Amazon Redshift aggregates query processing for concurrent sessions.



To view the cluster workload breakdown chart

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the cluster name from the list to open its details. The details of the cluster are displayed, which can include **Cluster performance**, **Query monitoring**, **Databases**, **Datashares**, **Schedules**, **Maintenance**, and **Properties** tabs.
3. Choose the **Query monitoring** tab for metrics about your queries.

4. In the **Query monitoring** section, choose **Database performance**, and choose **Cluster metrics**.

   The following metrics are graphed for the chosen time range as a stacked bar chart:

   - **Plan** time
   - **Wait** time
   - **Commit** time
   - **Execution** time

Analyzing query execution

You can analyze the execution details of a query to understand how it performed and identify potential areas for optimization. Analyzing a query provides insights into the query plan, including the steps involved, the time taken by each step, and the amount of data processed. Common use cases include troubleshooting slow-running queries, optimizing data distribution strategies, and identifying opportunities for query rewriting or indexing.

To analyze a query

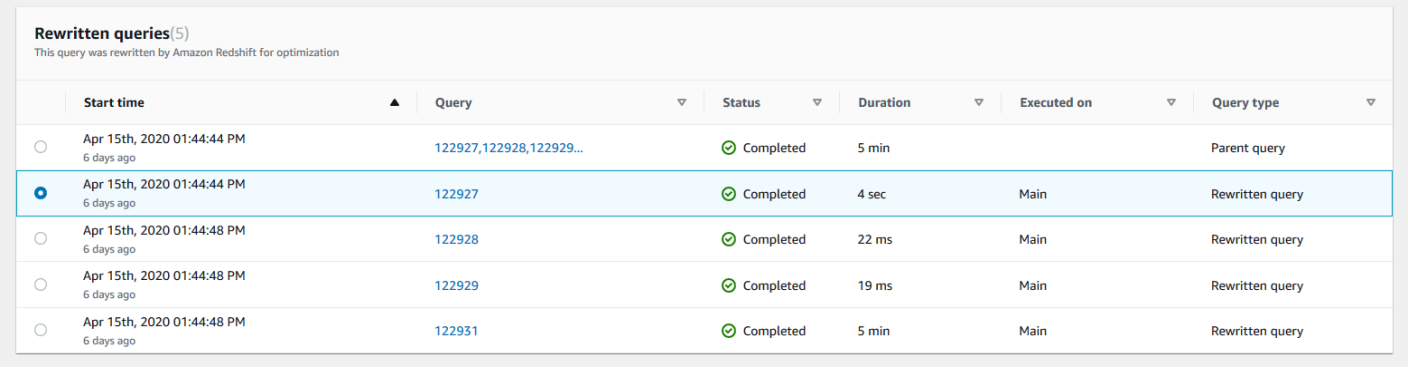
1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Queries and loads** to display the list of queries for your account. You might need to change settings on this page to find your query.
3. Choose the **Query** identifier in the list to display **Query details**.

   The **Query details** page includes **Query details** and **Query plan** tabs with metrics about the query.

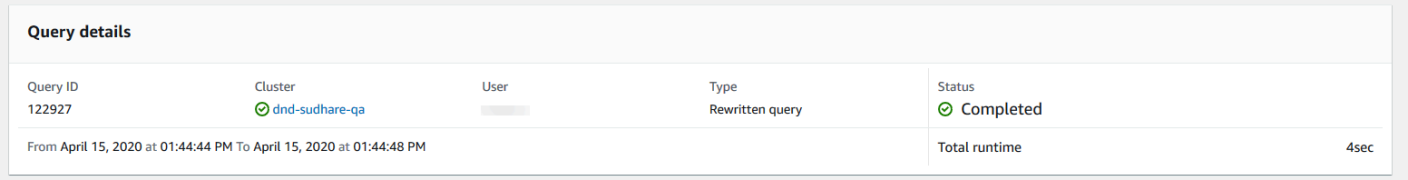
   > **Note**
   >
   > You can also navigate to the **Query details** page from a **Cluster details** page, **Query history** tab when you drill down into a query in a **Query runtime** graph.

The **Query details** page contains the following sections:

- A list of **Rewritten queries**, as shown in the following screenshot.



- A **Query details** section, as shown in the following screenshot.



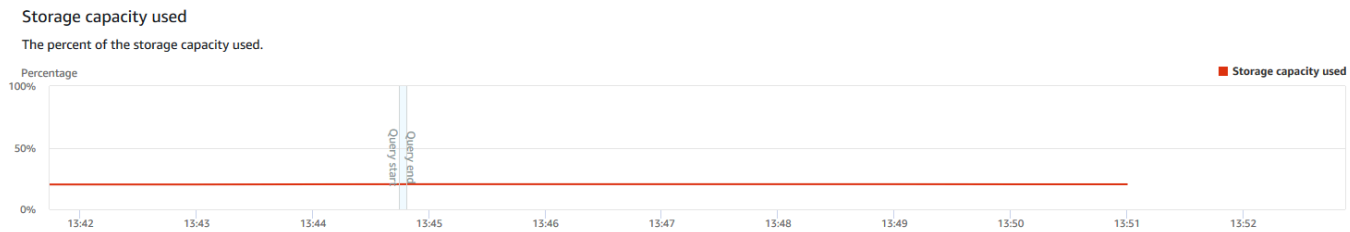
- A **Query details** tab that contains the **SQL** that was run and **Execution details** about the run.
- A **Query plan** tab that contains the **Query plan** steps and other information about the query plan. This table also contains graphs about the cluster when the query ran.
  - **Cluster health status**



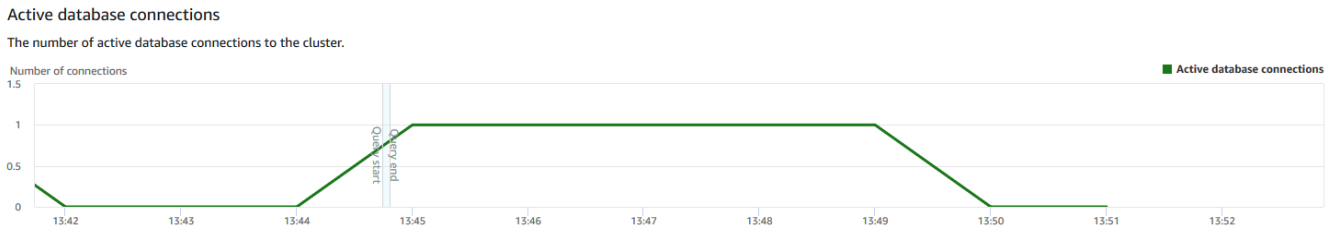
  - **CPU utilization**



  - **Storage capacity used**



- **Active database connections**



Creating an alarm

Alarms you create in the Amazon Redshift console are CloudWatch alarms. They are useful because they help you make proactive decisions about your cluster or serverless instance. You can set one or more alarms on any of the metrics listed in Performance data in Amazon Redshift. For example, setting an alarm for high `CPUUtilization` on a cluster node helps indicate when the node is overutilized. An alarm for high `DataStorage` would keep track of the storage space that your serverless namespace is using for your data.

From **Actions**, you can modify or delete alarms. You can also create a chime or slack alert to send an alert from CloudWatch to Slack or Amazon Chime by specifying a Slack or Amazon Chime webhook URL.

In this section, you can find how to create an alarm using the Amazon Redshift console. You can create an alarm using the CloudWatch console or any other way you work with metrics, such as with the AWS CLI or an AWS SDK.

To create a CloudWatch alarm with the Amazon Redshift console

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.

   If you're using Amazon Redshift Serverless, Choose **Go to Serverless** on the upper right of the dashboard.
2. On the navigation menu, choose **Alarms**, then choose **Create alarm**.
3. On the **Create alarm** page, enter the properties to create a CloudWatch alarm.
4. Choose **Create alarm**.

Ending a running query

You can also use the **Queries** page to end a query that is currently in progress.

> **Note**
>
> The ability to terminate queries and loads in the Amazon Redshift console requires specific permission. If you want users to be able to terminate queries and loads, make sure to add the `redshift:CancelQuerySession` action to your AWS Identity and Access Management (IAM) policy. This requirement applies whether you select the **Amazon Redshift Read Only** AWS managed policy or create a custom policy in IAM. Users who have the **Amazon Redshift Full Access** policy already have the necessary permission to terminate queries and loads. For more information about actions in IAM policies for Amazon Redshift, see Managing access to resources.

To end a running query

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Queries and loads** to display the list of queries for your account.
3. Choose the running query that you want to end in the list, and then choose **Terminate query**.

Performance metrics in the CloudWatch console

When working with Amazon Redshift metrics in the CloudWatch console, keep a couple of things in mind:

- Query and load performance data is only available in the Amazon Redshift console.
- Some Metrics in the CloudWatch have different units than those used in the Amazon Redshift console. For example, `WriteThroughput` is displayed in GB/s (as compared to Bytes/s in CloudWatch), which is a more relevant unit for the typical storage space of a node.

When working with Amazon Redshift metrics in the CloudWatch console, command line tools, or an Amazon SDK, keep these concepts in mind:

1. First, specify the metric dimension to work with. A dimension is a name-value pair that helps you to uniquely identify a metric. The dimensions for Amazon Redshift are `ClusterIdentifier` and `NodeID`. In the CloudWatch console, the `Redshift Cluster` and `Redshift Node` views are provided to easily select cluster and node-specific dimensions. For more information about dimensions, see Dimensions in the *CloudWatch Developer Guide*.
2. Then, specify the metric name, such as `ReadIOPS`.

The following table summarizes the types of Amazon Redshift metric dimensions that are available to you. Depending on the metric, data is available in either 1-minute or 5-minute intervals at no charge. For more information, see Amazon Redshift metrics.

CloudWatch namespace	Dimension	Description
`AWS/Redshift`	`NodeID`	Filters requested data that is specific to the nodes of a cluster. `NodeID` is either "Leader", "Shared", or "Compute-N" where N is 0, 1, ... for the number of nodes in the cluster. "Shared" means that the cluster has only one node, that is the leader node and compute node are combined.
`AWS/Redshift`	`ClusterIdentifier`	Filters requested data that is specific to the cluster. Metrics that are specific to clusters include `HealthStatus` , `MaintenanceMode` , and `DatabaseConnections` . General metrics for this dimension (for example, `ReadIOPS`) that are also metrics of nodes represent an aggregate of the node metric data. Take care in interpreting these metrics because they aggregate behavior of leader and compute nodes.

Working with gateway and volume metrics is similar to working with other service metrics. Many of the common tasks are outlined in the CloudWatch documentation, including the following:

- View available metrics
- Get statistics for a metric
- Creating CloudWatch alarms

Query profiler

This document describes Query profiler, a graphical tool for analyzing a query's components and performance.

Query profiler is a query monitoring and troubleshooting feature that's viewable through the Amazon Redshift console. It's helpful for analyzing query performance. Its primary purpose is to show a visual and graphical run order, execution plan, and statistics about a query, and make them easier to understand and troubleshoot. Query profiler helps you analyze the following types of query components:

- **Child queries** – A child query is a portion of work from a query. Amazon Redshift may split a query into multiple child queries if it's more efficient than processing it as a single large query. In the profiler you can see the properties of each child query. A child query is made up of streams and additional sub-components.

  The types of child query that Query profiler shows typically include the following:

  - **Temp table query**: The text of this child query starts with the `CREATE TEMP TABLE` command. This child query creates temporary tables for other child queries to process.
  - **Statistics query**: Query profiler adds the following comment to the start of this child query to help identify it:

    ```
    -- collect statistics of child query queryID
    ```

    This child query collects information that the Amazon Redshift query engine uses to optimize performance.

  > **Note**
  >
  > Query Profiler shows the query that the user provides as the final child query that Amazon Redshift runs.

- **Streams** – A stream is a collection of segments parceled over available compute-node slices. Each child query is made up of one or more segments. In Query profiler, you can see the properties of each stream, such as its execution time. By glancing through the list of streams, it's likely that you can quickly find performance bottlenecks.
- **Segments** – A segment is a combination of several steps that a single process can run. A segment is also the smallest compilation unit executable by a compute node slice. A slice is the unit of

parallel processing in Amazon Redshift. The segments in a stream run in parallel. Query profiler doesn't show segments graphically, but you can access the segment information for a step in that step's details pane.

- **Steps** – Each segment is made up of a collection of steps. A step is a piece of work in a query. Steps can include a *hashjoin*, for example, or a *scan*, which is the reading of records from a table.

For more information about streams, segments, and steps, see Query planning and execution workflow in the *Amazon Redshift Database Developer Guide*.

Query profiler displays information returned by the SYS_QUERY_HISTORY, SYS_QUERY_DETAIL, SYS_QUERY_EXPLAIN, and SYS_CHILD_QUERY_TEXT views. For more information about these views, see SYS_QUERY_HISTORY, SYS_QUERY_DETAIL, SYS_QUERY_EXPLAIN, and SYS_CHILD_QUERY_TEXT in the *Amazon Redshift Database Developer Guide*.

Query profiler only displays query information for queries that have recently run on the database. A query that completes using prepopulated cache data rather than running against the database won't have a query profile if information wasn't previously available for it. This is because Amazon Redshift doesn't generate a query plan for it.

Prerequisites for using Query profiler

The SYS monitoring views are designed for ease of use and reduced complexity, providing a complete array of metrics for effective monitoring and troubleshooting. The SYS monitoring views also guarantee query history for the past seven days regardless of the size or activity of the cluster. Users only have visibility for queries that they ran, while super users have visibility for queries from all users.

Your IAM user account or role needs permissions to access the **Query and database monitoring** section of the console. This section describes how to add permissions to a user account or role.

Use the following policy to add minimum permissions to your IAM user account or role:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "redshift:DescribeClusters",
                "redshift-serverless:ListNamespaces",
                "redshift-serverless:ListWorkgroups",
                "redshift-data:ExecuteStatement",
                "redshift-data:DescribeStatement",
                "redshift-data:GetStatementResult"
            ],
            "Resource": [
                "arn:aws:redshift-serverless:us-east-1:111122223333:*",
                "arn:aws:redshift:us-east-1:111122223333:*"
            ]
        }
    ]
}
```

Topics

- Granting query monitoring permissions for a role
- Granting query monitoring permissions for a user
- Temporary credentials using your IAM identity

Granting query monitoring permissions for a role

Users with a role that has the `sys:monitor` permission can view all queries. Users with a role that has the `sys:operator` permission can cancel queries, analyze query history, and perform vacuum operations.

To grant query monitoring permission for a role

1. Use the following command to provide system monitor access, where *role-name* is the name of the role for which you want to provide access.

   ```
   grant role sys:monitor to "IAMR:role-name";
   ```

2. (Optional) Use the following command to provide system operator access, where *role-name* is the name of the role for which you want to provide access.

   ```
   grant role sys:operator to "IAMR:role-name";
   ```

Granting query monitoring permissions for a user

Users with the `sys:monitor` permission can view all queries. Users with the `sys:operator` permission can cancel queries, analyze query history, and perform vacuum operations.

To grant query monitoring permission for a user

1. Use the following command to provide system monitor access, where *`user-name`* is the name of the user for which you want to provide access.

   ```
   grant role sys:monitor to "IAMR:user-name";
   ```

2. (Optional) Use the following command to provide system operator access, where *`-name`* is the name of the user for which you want to provide access.

   ```
   grant role sys:operator to "IAMR:user-name";
   ```

Temporary credentials using your IAM identity

This option is only available when connecting to a cluster. With this method, Query Profiler maps a user name to your IAM identity and generates a temporary password to connect to the database as your IAM identity. A user using this method to connect must be allowed IAM permission to `redshift:GetClusterCredentialsWithIAM`. To prevent users from using this method, modify their IAM user or role to deny this permission.

Accessing Query profiler in the Amazon Redshift console to analyze a query

You can access Query Profiler for either Amazon Redshift serverless or Amazon Redshift provisioned. See the following sections for details:

Topics

- Accessing Query profiler in the Amazon Redshift console for Amazon Redshift Serverless
- Accessing Query profiler in the Amazon Redshift console for Amazon Redshift Provisioned

Accessing Query profiler in the Amazon Redshift console for Amazon Redshift Serverless

To access Query profiler for Amazon Redshift Serverless, do the following:

- Open the Amazon Redshift serverless console.

- On the navigation pane, under **Monitoring**, choose **Query and database monitoring**.
- Choose a workgroup.
- Choose **Queries and database monitoring**.
- Choose a query.
- Choose the **Query plan** tab in the **Query details** page.

If a query plan is available, you will see a list of child queries. Choose a query to view it in Query profiler.

Accessing Query profiler in the Amazon Redshift console for Amazon Redshift Provisioned

To access Query profiler for Amazon Redshift Provisioned, do the following:

- Open the the Amazon Redshift Provisioned clusters dashboard.
- Choose a cluster
- Choose **Query monitoring**.
- Connect to a database
- Choose **Queries and database monitoring.**
- Choose a query.

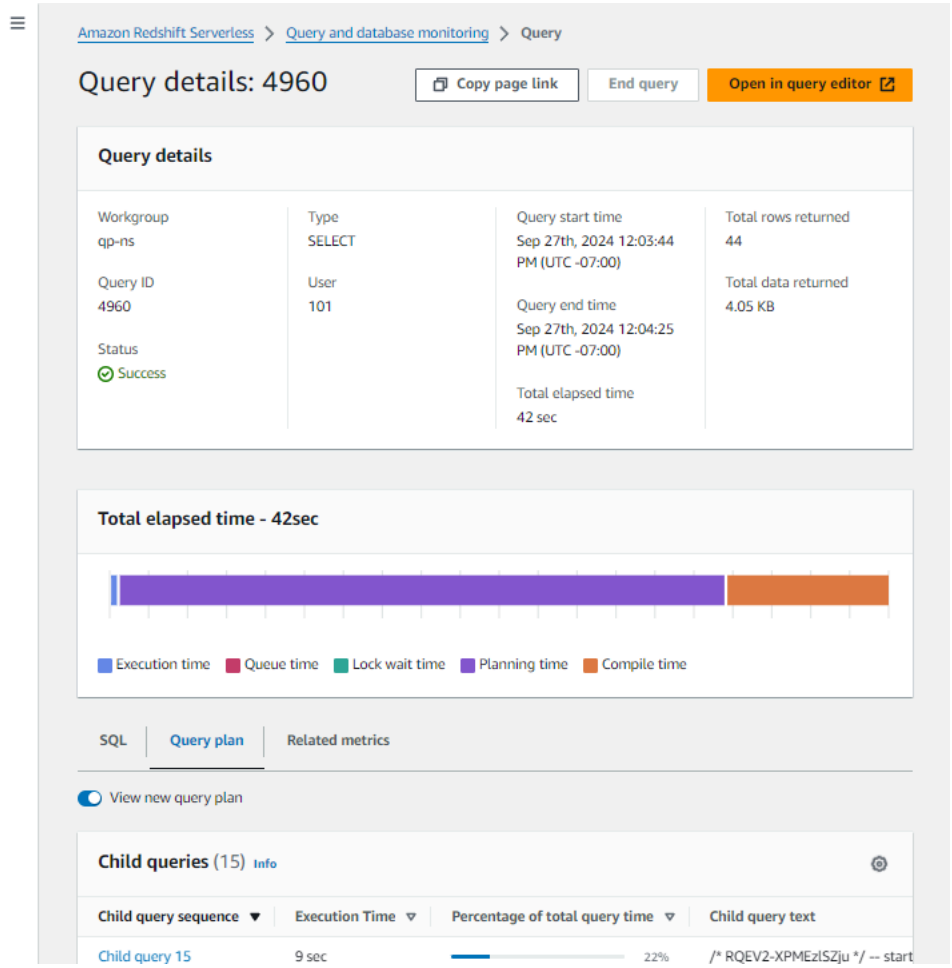
If a query plan is available, you will see a list of child queries. Choose a query to view it in Query profiler.

Query profiler user interface

Query profiler uses the following pages to display information about your query:

- **Query details page** – This page displays statistics and child queries for your query.
- **Child query page** – This page displays statistics, streams, and a visual representation of the execution plan for a child query. The console displays this page when you choose a child query from the **Child queries** list in the **Query and database monitoring** page.

Query details page



The Query details page has the following components:

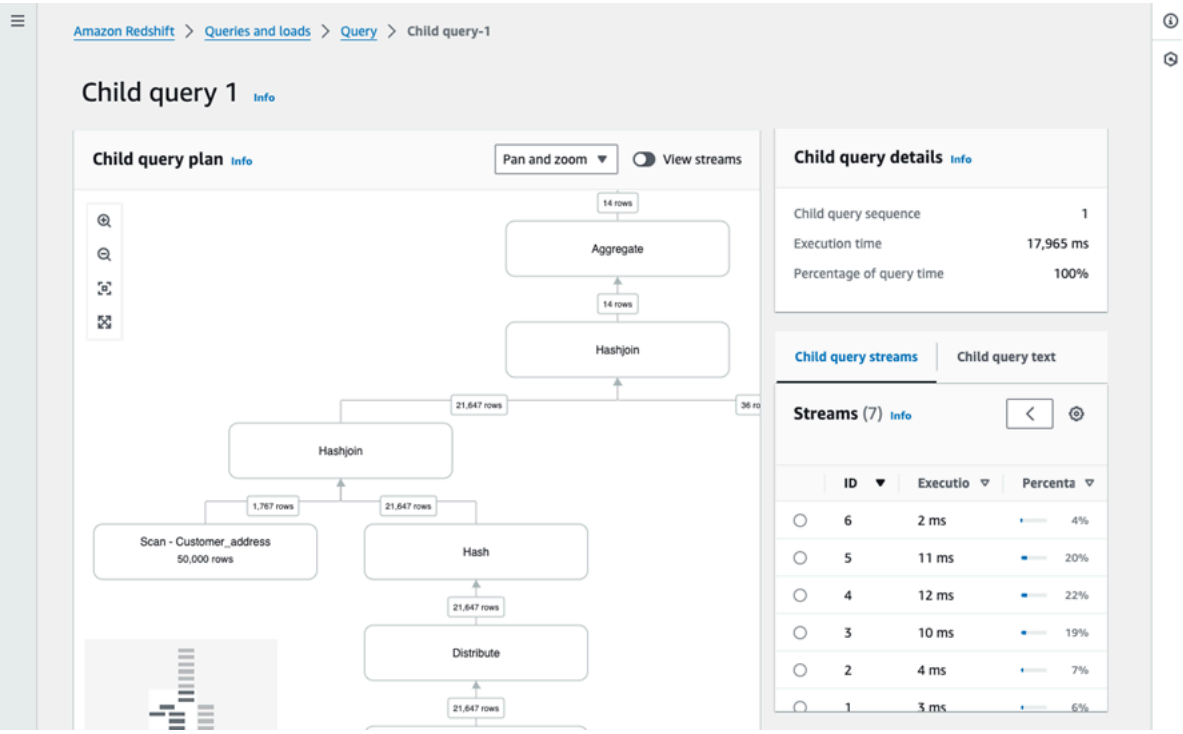
- **Top pane** – The pane at the top of the page shows details about the query, such as the status and type. For information about the source of the information that the top pane shows, see SYS_QUERY_HISTORY in the *Amazon Redshift Database Developer Guide*.
- **SQL tab** – This tab of the bottom panel shows the SQL text for the original user query.
- **Query plan tab** – This tab of the bottom panel shows a list of the child queries that Amazon Redshift uses to prepare data and statistics for the user query. By default, the **Child queries** list shows information and aggregated statistics about each child query. For information about the source of the information that this page shows, see SYS_QUERY_DETAIL in the *Amazon Redshift Database Developer Guide*.

  You can add or remove columns from the **Child queries** list using the Preferences menu.

- **Related metrics** – This tab of the bottom panel shows the following CloudWatch metrics for the query:
  - **RPU capacity used** (for serverless workgroups): The compute capacity used by the query, measured in Redshift Processing Units (RPUs). For more information, see Compute capacity for Amazon Redshift Serverless.
  - **Cluster health status**, **CPU Utilization**, **Storage capacity used** (for provisioned clusters): The status and system resources the query used.
  - **Active database connections**: the `DatabaseConnections` metric for the query.

  For more information about CloudWatch metrics, see Performance data in Amazon Redshift.

Child query page



The Child query page has the following components:

- **Child queries pulldown** – This control shows the sequence name and the execution time for each child query. You can navigate to other child queries by selecting them in this control.
- **Side panel** – This panel contains tabs for displaying the child query streams and the child query text.
- **Child query streams tab** – This tab of the top panel displays the following:

- **Streams** – This pane shows the list of streams in the child query. This pane shows information and aggregated data about the streams that Amazon Redshift uses to optimize the query. For information about the details in this pane, see SYS_QUERY_DETAIL in the *Amazon Redshift Database Developer Guide*.

  You can add or remove columns from the **Streams** pane using the Preferences menu. You access the Preferences menu using the gear icon in the **Streams** pane.

- **Child query pane** – A graphical representation of the steps in the child query. For information about the **Child query** pane, see Child query pane following.

- **Child query text tab** – This tab of the top panel shows the SQL code for the child query.
- **Child query details pane** – This pane in the right panel shows details about the child query. For information about the details in this pane, see SYS_QUERY_DETAIL in the *Amazon Redshift Database Developer Guide*.
- **Stream details pane** – When you choose a stream in the **Streams** pane, the **Stream details** pane shows information about the stream. For information about the details in this panel, see SYS_QUERY_DETAIL in the *Amazon Redshift Database Developer Guide*.
- **Step details pane** – When you choose a step in either the **Streams** pane or the graph in the **Child query plan**, the **Step details** pane shows information about the step, For information about the details in this pane, see SYS_QUERY_DETAIL in the *Amazon Redshift Database Developer Guide*.:

Child query pane

Query profiler displays the child query in the **Child query** pane as a graphical representation of the steps in the selected child query.

The **Child query** pane displays the run order and the relationships between the steps. For example, if a step joins the output from two other steps, the **Child query** pane shows the step as a tree node with two nodes feeding into it:



The **Child query** pane doesn't show the streams that contain the steps by default. To show the streams that Amazon Redshift uses to logically partition the steps in the child query, choose **View**

streams. When you choose **View streams**, the **Child query** pane shows the steps contained by the query's streams.

The **Child query** pane doesn't show segment information. To see the segment for a step, choose the step. The **Step details** pane then shows the segment for the step.

Navigation in the Child query pane

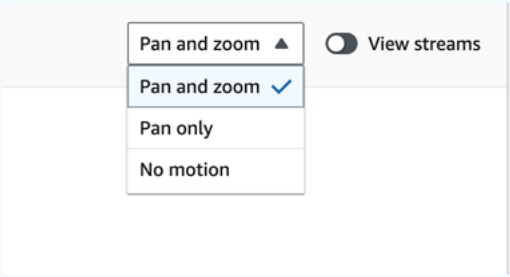
In the **Child query** pane, you can choose steps to see detailed information about them. You can also pan and zoom the workspace to better visualize the steps in your query plan.

You can choose nodes, pan, and zoom in the **Child query** plan using the following methods:

- **Using the mouse** – You can choose nodes, click and drag the workspace to pan, and hold **`Ctrl`** (Windows) or **`CMD`** (Mac) and mousewheel to zoom. Choosing a node causes the workspace to zoom and pan to highlight that node. If you choose a stream in the workspace, that stream highlights in the **Streams** list. If you choose a step in the workspace, the **Step details** pane shows information about that step.
- **Using the zoom and fit controls in the upper left of the workspace** – These controls let you zoom in, zoom out, zoom to fit the entire workspace, and enter fullscreen mode. When you zoom to fit the entire query plan, the workspace centers the query plan both horizontally and vertically.
- **Using the minimap in the lower right of the workspace** – You can pan or zoom the workspace by using the minimap control in the lower left corner of the workspace.
- **Choosing a stream in the Streams pane** – If you choose a stream in the **Streams** pane, the workspace pans and zooms to show the selected stream, and shows information about the stream in the **Stream details** pane.
- **Choosing a step in the Stream details pane** – If you choose a step in the **Stream details** pane, the workspace pans and zooms to show the selected step, and shows information about the step in the **Step details** pane.

> **Note**
>
> When you choose a step, either in the workspace or a different pane, the workspace attempts to zoom and pan such that the selected step is best visible.
> When you choose a stream or step from either the workspace or another pane, the workspace will only zoom and pan to that stream or step if you have **Pan & Zoom** selected

in the upper right control in the workspace. You can restrict this behavior to panning and zooming, panning only, or no movement by choosing the appropriate setting in the pulldown.



Troubleshooting queries with Query profiler

If you're troubleshooting a query, you can choose a child query to determine which stream uses the highest **Percentage of total query time**. This is a quick way to determine which part of your query should be analyzed further.

After you know which child query is taking the most time, view its steps to see which join or scan might be causing slow performance.

Query and Database Monitoring

This document describes the Query and Database Monitoring page, an AWS Management Console feature for analyzing the performance of an Amazon Redshift provisioned cluster or serverless workgroup, and the queries run against them.

You can investigate the following scenarios using the Queries and Database Monitoring page:

- Monitor data warehouse metrics during a specified time period
- How a query contribues to the overall performance of a data warehouse
- See a breakdown of a query run time into its life cycle events such as lock wait time, compile time, and execution time
- Which users run the most resource-intensive queries during a specified time period
- Monitor how patch events affect query performance

Topics

- Permissions
- Query and Database Monitoring Console

Permissions

The privileges of the AWS account you use to access the console affects which queries the **Query and Database Monitoring** page shows. By default, you can only view your own queries. To view queries owned by other users, grant the SYS:MONITOR role to your account. To allow a user to end running queries from the **Query and Database Monitoring** page, grant the user the SYS:OPERATOR privilege.

To automatically assign the `sys:monitor` role to an IAM user or role for Amazon Redshift Serverless or provisioned, run the following commands:

```
create role monitor;
grant role sys:monitor to role monitor;
```

To update the IAM role used for query monitoring, do the following:

1. Choose the **Tags** tab.
2. Choose **Manage tags**.
3. Add a tag with key **`RedshiftDbRoles`** and value **`monitor`**.
4. Save changes

To add database credentials to a user, run the following command:

```
grant role sys:monitor to <username>
```

For information about using the GRANT command, see GRANT in the *Amazon Redshift Database Developer Guide*.

To use query monitoring, your IAM user needs permissions to access the Amazon Redshift data plane. Ensure that your IAM user has the following permissions in their permissions policy:

```
{
    "Sid": "DataAPIPermissions",
    "Action": [
```

```
        "redshift-data:ExecuteStatement",
        "redshift-data:CancelStatement",
        "redshift-data:GetStatementResult",
        "redshift-data:DescribeStatement",
        "redshift-data:ListDatabases"
    ],
    "Effect": "Allow",
    "Resource": "arn:aws:redshift-serverless:us-
west-2:123456789012:workgroup/01234567-89ab-cdef-0123-456789abcdef"
},
```

Temporary credentials using your IAM identity

This option is only available when connecting to a cluster. With this method, Query and Database Monitoring maps a user name to your IAM identity and generates a temporary password to connect to the database as your IAM identity. A user using this method to connect must be allowed IAM permission to `redshift:GetClusterCredentialsWithIAM`. To prevent users from using this method, modify their IAM user or role to deny this permission.

Query and Database Monitoring Console

This section describes using the Query and Database Monitoring console page.

You can use the Query and Database Monitoring console to quickly get an overview of your data warehouse's performance. You can monitor your data warehouse's performance over time, and examine the performance of a data warehouse's provisioned clusters or individual queries, so as to best identify bottlenecks and other areas to improve.

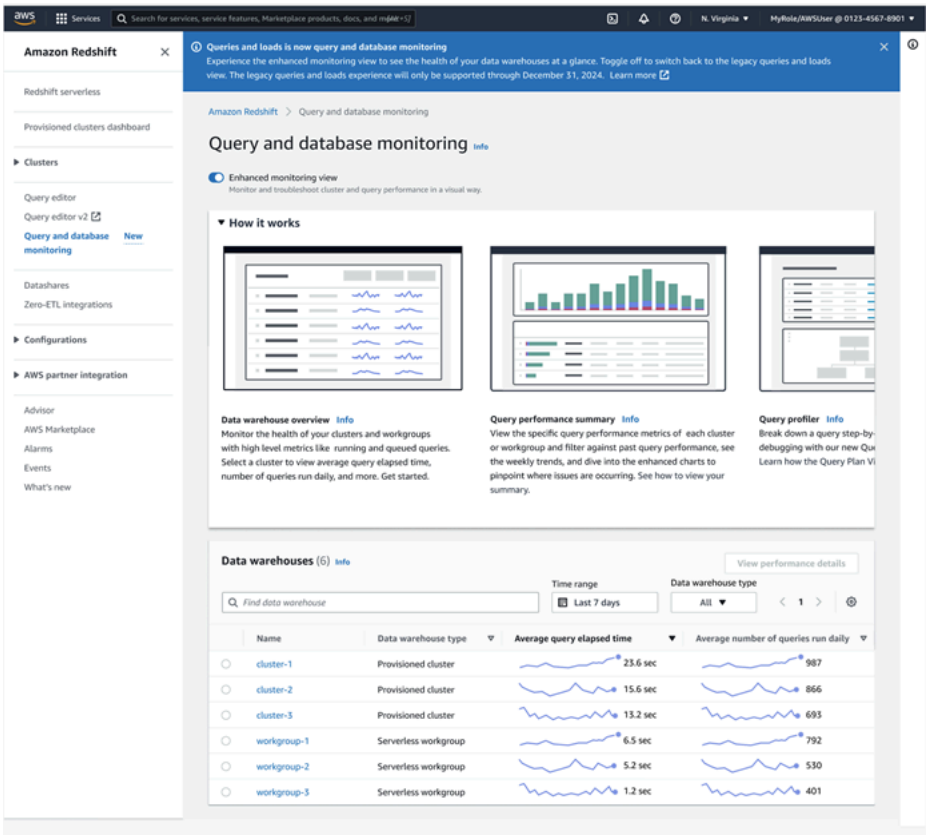
The **Query and Database Monitoring** page has the following features:

- **Increased security** — You need elevated privileges to monitor queries for other users. For more information, see Permissions.
- **Seven-day query history** — Guaranteed access to seven days of query history
- **Query monitoring** — You can monitor queries in provisioned clusters and serverless workgroups at the user query level.
- **Query trend analysis** — You can compare the performance of similar queries that match specified criteria.

To access the Query and Database Monitoring page, do the following:

1. Sign in to the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. Choose **Query and database monitoring** from the navigation pane.

The **Query and database monitoring** console page appears as follows:



The **Query and database monitoring** page has the following components:

- **Data warehouse overview** — Monitor the average query performance for your provisioned clusters and serverless workgroups. You can quickly see performance issues for a specific cluster or workgroup by examining the statistics on this page for spikes or periods of high activity.
- **Query performance summary** — Monitor the average query performance for a specific provisioned cluster or serverless workgroup. You can also acccess the **Query performance summary** page by clicking on one of the clusters or workgroups in the **Data warehouse overview** list.

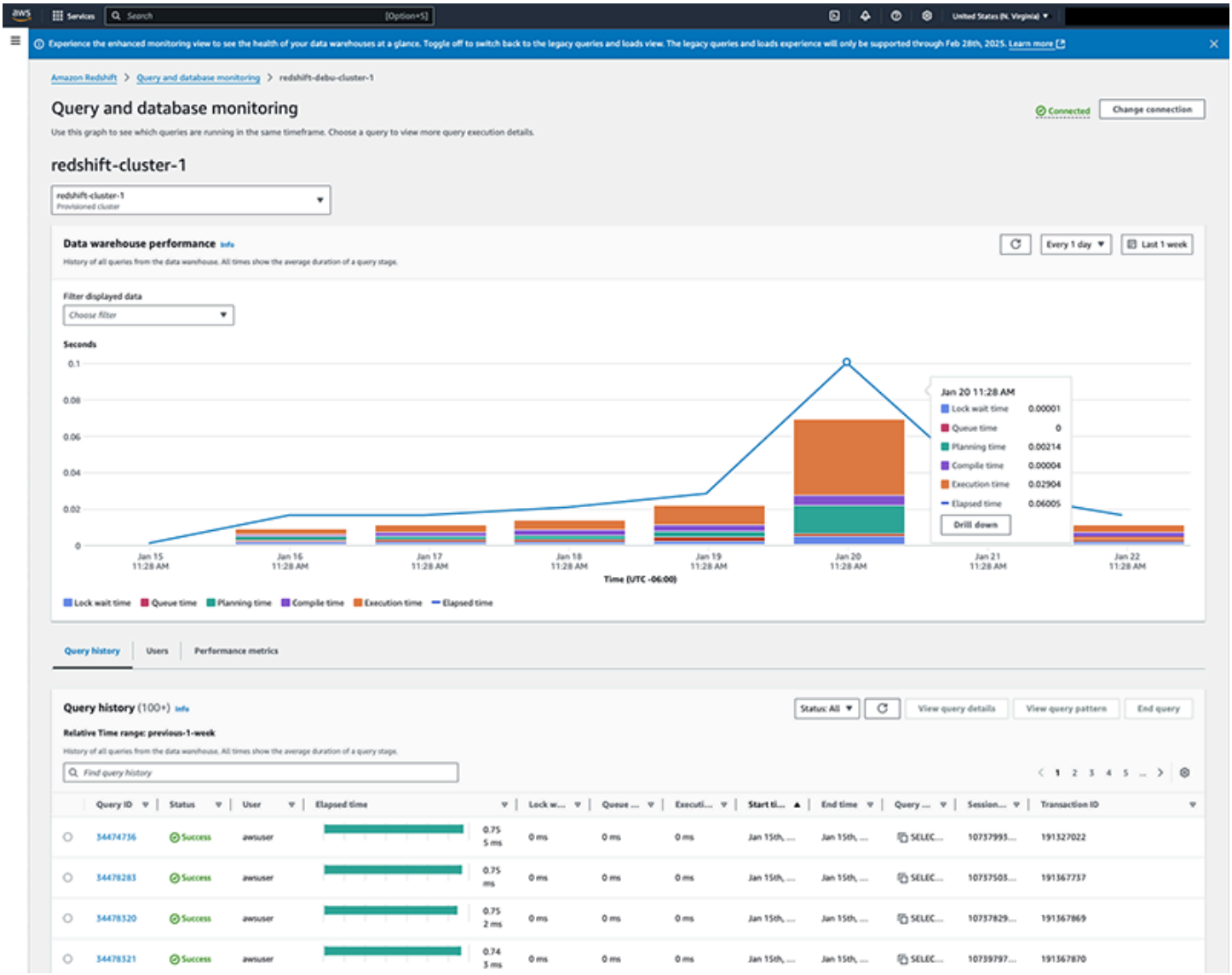
Topics

- Query performance summary
- Query details

- Query pattern

Query performance summary

When you choose **Query performance summary** from the **Query and database monitoring** page, or a cluster or workgroup from the **Data warehouse overview** page, the console shows a summary of the performance for an individual provisioned cluster or serverless workgroup.



This page has the following components:

- **Cluster or workgroup dropdown** — Choose the cluster or workgroup you want to analyze.
- **Data warehouse performance** — This pane shows a history of the cluster or workgroup within the specified time period, showing the amount of time spent on each phase of queries. If you see

a spike in a certain query phase, such as the increased planning and execution time on January 20 in the preceding graphic, you can use this information to identify issues with the performance of your queries. The default time period shown is the last seven days, but you can adjust the time period according to your analysis needs.

- **Query history** — This pane shows a history of the performance of every query run within the specified filters. You can use this information to troubleshoot the performance of an individual query. To further analyze a specific query, you can choose it from this graph or from the **Data warehouse performance** graph.

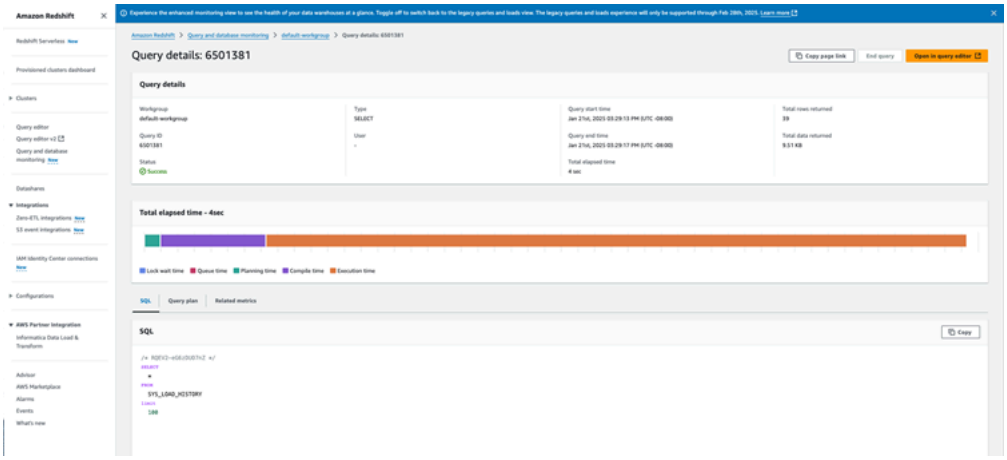
  When you hover over a data point on the chart, a popup appears. This popup displays the breakdown of time metrics for that data point. If a data point contains query data, you can choose **Drill down** to update the time range of the chart to the next smallest time unit of the data point. These time units are as follows:

  - Day
  - Hour
  - 15 minutes
  - 5 minutes
  - 1 minute

  For example, if you choose **Drill down** on a data point, the range of the graph changes to one day. Choosing **Drill down** again changes the range of the graph to one hour.
- **Query profiler** — A graphical tool for monitoring query performance. For more information, see Query profiler.

Query details

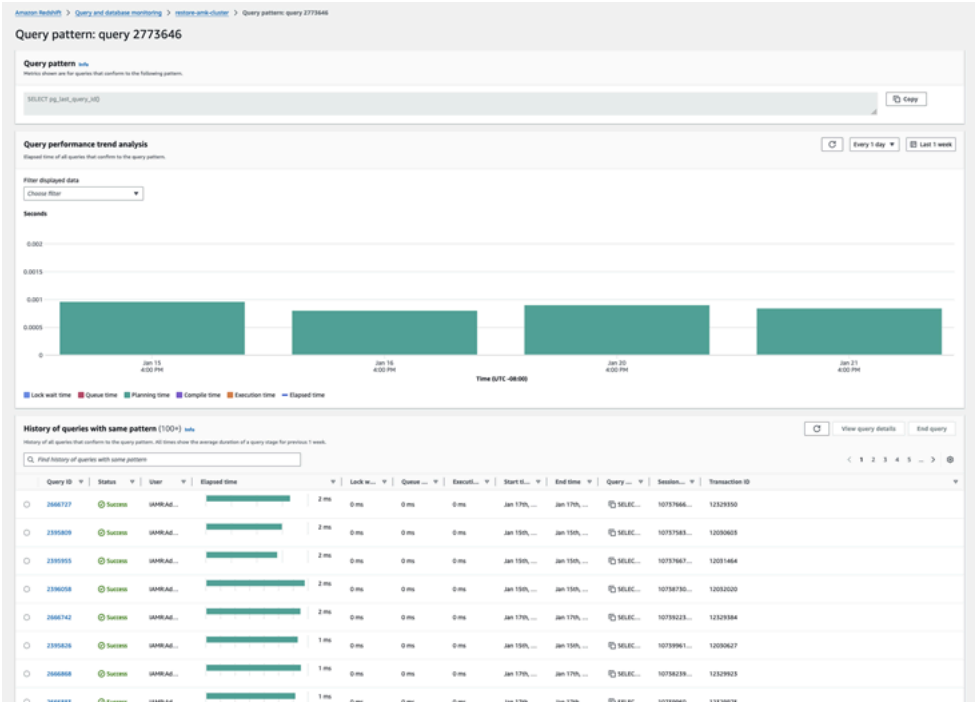
When you choose a query from the **Data warehouse performance** or **Query history** pane of the cluster or workgroup detail page, the **Query details** page opens.



For information about the **Query details** page, see Query details page.

Query pattern

You can see a history of queries with the same pattern by choosing the **View query pattern** button in the **Query history** pane of the **Query performance summary** page. The **Query pattern** page shows all of the queries from the past week that are retrieved by a SQL statement that you specify.



The **Query pattern** page has the following components:

- **Query pattern** — The SQL statement that retrieves the queries being analyzed.

- **Query performance trend analysis** — A bar graph showing the elapsed time of all the queries that the query pattern selects. The results are grouped by day.
- **History of queries with the same pattern** — The time taken for each phase of the queries that the query pattern selects.

Using the **Query pattern** page, you can gain the following insights:

- Trends for queries that run at a specific time every day
- Spikes in run time for queries that you run regularly against the data warehouse.

Sys View-based Queries and Database Monitoring

This document describes the SYS views that provide data for the Queries and Database Monitoring page in the Amazon Redshift Console, a tool for analyzing a query's components and performance. For information about the Queries and Database Monitoring page, see Query and Database Monitoring.

The **Queries and Database Monitoring** page has a functionality that displays information provided by SYS views. The console view includes the query profiler, which shows the graphical execution plan of a query. To switch to the SYS-based view, follow the steps here to grant the right access and permissions for the new **Queries and Database Monitoring** page.

The SYS-based view feature of the **Queries and Database Monitoring** page has the following functionality:

- **Increased security** — You need elevated privileges to monitor queries for other users
- **Seven-day query history** — Guaranteed access to seven days of query history
- **Query profiler** — A graphical tool for monitoring query performance. For more information, see Query profiler

By default, you can only view your own queries. To view queries owned by other users, grant the `SYS:MONITOR` role to your account. To allow a user to end running queries, grant the user the `SYS:OPERATOR` privilege.

To grant the privilege to view queries owned by all users to a database user or role, run the following commands:

```
grant role sys:monitor to "IAM:role-name";
grant role sys:monitor to "IAM:user-name";
```

To automatically assign the `sys:monitor` role to an IAM user or role for Amazon Redshift Serverless or provisioned, run the following commands:

```
 create role monitor;
grant role sys:monitor to role monitor;
```

To update the IAM role used for query monitoring, do the following:

1. Choose the **Tags** tab.
2. Choose **Manage tags**.
3. Add a tag with key **`RedshiftDbRoles`** and value **`monitor`**.
4. Save changes

To add database credentials to a user, run the following command:

```
grant role sys:monitor to <username>
```

Permissions

To use query monitoring, your IAM user needs permissions to access the Amazon Redshift data plane. Ensure that your IAM user has the following permissions in their permissions policy:

```
{
    "Sid": "DataAPIPermissions",
    "Action": [
        "redshift-data:ExecuteStatement",
        "redshift-data:CancelStatement",
        "redshift-data:GetStatementResult",
        "redshift-data:DescribeStatement",
        "redshift-data:ListDatabases"
    ],
    "Effect": "Allow",
    "Resource": "arn:aws:redshift-serverless:us-
west-2:123456789012:workgroup/01234567-89ab-cdef-0123-456789abcdef"
},
```

Connect to the database

Before using the enhanced query monitoring feature, you must first connect to your database for access to SYS view-based information. To connect to the database, use one of the following credentials:

- Username and password
- Temporary credentials associated with your IAM role
- A database user

Note the following about using enhanced query monitoring:

- For provisioned clusters, you must connect to a database, because enhanced query monitoring uses SYS views. These views have increased security, and require elevated privileges to access data about queries owned by other users.
- When using the SYS view-based queries and database monitoring page, only your `user_id` is visible if your user account doesn't have the database superuser role. Usernames are hidden from non-superusers.
- As part of the sys view-based queries and database monitoring page experience, the query execution process ID (`p_id`) appears under the column heading `session_id`.

Tracks for Amazon Redshift provisioned clusters and serverless workgroups

When Amazon Redshift releases a new version, it updates the version of your Amazon Redshift data warehouse (serverless workgroup or provisioned cluster). You can control whether your data warehouse is updated to the most recent release or to the previous certified release.

The serverless workgroup or provisioned cluster's track determines which released version is applied during a version update. Amazon Redshift updates provisioned clusters during the specified maintenance window, and usually updates serverless workgroups during idle periods. For details of when Redshift Serverless updates workgroups, see Updating serverless workgroups.

When Amazon Redshift releases a new version, that version is assigned to the *current* track, and the previous version is assigned to the *trailing* track. To set the track for your data warehouse, specify one of the following values:

- **Current** – With the **Current** track, you get the most up-to-date certified release version with the latest features, security updates, and performance enhancements.
- **Trailing** – With the **Trailing** track, you will be on the previous certified release.

For example, suppose that your serverless workgroup is currently running version 1.0.2762 and Amazon Redshift releases Redshift Serverless version 1.0.3072. If your track value is **Current**, your workgroup is updated to version 1.0.3072 (the latest release). If you set the track value to **Trailing**, your workgroup is updated when the next trailing track version is released.

With the trailing track feature, you have the option to run a subset of Amazon Redshift data warehouses in the trailing track. This allows for 1-6 weeks of testing and integration validation on data warehouses set to the Current track before applying the release to data warehouses on the Trailing track. By default, Amazon Redshift creates all clusters and workgroups on the Current track to take advantage of the most up-to-date, certified release. However, using the Amazon Redshift trailing track in your production environment, and the current track in your testing and development environment, gives you additional diligence and time to evaluate the latest release. The Trailing track ensures maximum stability, making it ideal for mission-critical workloads in production environments.

> **Note**
>
> The trailing track version may be the same as the current track version for short periods of time. This happens when the current track hasn't advanced to the next version. Normally, the current track version is ahead of the trailing track version.

Switching between tracks

Changing tracks for an Amazon Redshift resource is generally a one-time decision. You should exercise caution in changing tracks. For information about which features are in which data warehouse versions, see Cluster versions for Amazon Redshift.

If you change the track from **Trailing** to **Current**, we will update the data warehouse to the **Current** track release version. If you change the data warehouse's track to **Trailing**, we will update your data warehouse as follows:

- For serverless workgroups, we update your data warehouse's version during an idle period. For more details about how Redshift Serverless updates your workgroup's version, see Updating serverless workgroups.
- For provisioned clusters, we won't update your data warehouse until there is a new release after the **Current** track release version.

Tracks and restore

For serverless workgroups, a snapshot inherits the target Amazon Redshift data warehouse's track. For example, if you create a snapshot for a workgroup set to Trailing track, and apply that snapshot to a workgroup set to Current track, the workgroup will have a track setting of Current.

For provisioned clusters, a snapshot inherits the source Amazon Redshift data warehouse's track. If you change the source data warehouse's track after you take a snapshot, the snapshot and the source data warehouse are on different tracks. When you restore from the snapshot, the new data warehouse will be on the track that was inherited from the snapshot source. You can change the track after the restore operation completes.

Resizing a data warehouse doesn't affect its track.

Updating serverless workgroups

When a new version beomes available for a workgroup's chosen track, Amazon Redshift Serverless typically applies the update during an idle period as long as there is no pending track update request. If the workgroup doesn't experience an idle period within 14 days, Redshift Serverless forces the version update.

Redshift Serverless only updates your workgroup to the next higher version. Redshift Serverless doesn't skip intermediate versions or downgrade workgroups, even if the version for the selected workgroup's track is less than the workgroup's current version. Your workgroup will not receive any major version upgrades until and unless the `Trailing` track catches up.

For example, suppose the `Current` track is version 186, and the `Trailing` track version is 185. If you have a workgroup that has a `Track` value of `Current`, whose version is 186, if you change the `Track` value to `Trailing`, Redshift Serverless won't downgrade the workgroup's version to 185. In this scenario, Redshift Serverless keeps the workgroup on version 186 until the `Trailing` track version is equal to or higher than 186.

If a track change is pending, Redshift Serverless doesn't update the workgroup to the next major version in the exisiting track until it applies the track change. After the track change is complete, Redshift Serverless evaluates the conditions for updating the workgroup to the appropriate version under the new track.

For example, if your workgroup is set to the `Current` track, and the current track is 186, and you change your workgroup to the `Trailing` track, Redshift Serverless won't update the workgroup until after it applies the track change, and after the `Trailing` version is updated to a version equal to or higher than 186.

> **Note**
>
> Any existing operation on a workgroup, such as restoring from a snapshot, changing the KMS key, or resizing, only happens on the existing track. Redshift Serverless does not use the pending track for serverless operations.

If you have a pending track switch request, you can cancel the request by setting the `track` parameter back to its original value using UpdateWorkgroup.

Managing versions

A track is a series of releases. You can decide if your Amazon Redshift data warehouse is on the **Current** track or the **Trailing** track. If you put your data warehouse on the **Current** track, it will always be upgraded to the most recent release version. If you put your resource on the **Trailing** track, it will always run the release version that was released immediately before the most recently released version.

For provisioned clusters, the **Release status** column in the Amazon Redshift console list of Amazon Redshift data warehouses indicates whether one of your resources is available for upgrade.

Determining the workgroup or cluster version

You can determine the Amazon Redshift serverless workgroup version or provisioned cluster version engine with the Amazon Redshift console.

Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.

Serverless workgroups

For serverless workgroups, on the navigation menu, choose **Workgroups**, then choose the workgroup name from the list to open its details. The details of the workgroup are displayed.

Provisioned clusters

For provisioned clusters, on the navigation menu, choose **Clusters**, then choose the cluster name from the list to open its details.

The details of the cluster are displayed, which can include **Cluster performance**, **Query monitoring**, **Databases**, **Datashares**, **Schedules**, **Maintenance**, and **Properties** tabs. Choose the **Maintenance** tab for more details.

In the **Maintenance** section, find **Current cluster version**.

> **Note**
>
> For provisioned clusters, the console displays the version information in one field, but it's two parameters in the Amazon Redshift API. These parameters are

`ClusterVersion` and `ClusterRevisionNumber`. For more information, see Cluster in the *Amazon Redshift API Reference*.

Zero-ETL integrations

Zero-ETL integration is a fully managed solution that makes transactional and operational data available in Amazon Redshift from multiple operational and transactional sources. With this solution, you can configure an integration from your source to an Amazon Redshift data warehouse. You don't need to maintain an extract, transform, and load (ETL) pipeline. We take care of the ETL for you by automating the creation and management of data replication from the data source to the Amazon Redshift cluster or Redshift Serverless namespace. You can continue to update and query your source data while simultaneously using Amazon Redshift for analytic workloads, such as reporting and dashboards.

With zero-ETL integration you have fresher data for analytics, AI/ML, and reporting. You get more accurate and timely insights for use cases like business dashboards, optimized gaming experience, data quality monitoring, and customer behavior analysis. You can make data-driven predictions with more confidence, improve customer experiences, and promote data-driven insights across the business.

The following sources are currently supported for zero-ETL integrations:

- Amazon Aurora MySQL (AMS)
- Amazon Aurora PostgreSQL (APG)
- Amazon DynamoDB
- Amazon RDS for MySQL
- Amazon RDS for Oracle
- Amazon RDS for PostgreSQL
- Oracle Database@AWS
- Applications including Salesforce, Salesforce Marketing Cloud Account Engagement, SAP, ServiceNow, Instagram ads, Meta ads, and Zendesk
- Self-Managed MySQL, PostgreSQL, SQL Server, and Oracle

To create a zero-ETL integration, you specify an integration source and an Amazon Redshift data warehouse as the target. After an initial data load, the integration replicates data from the source to the target data warehouse. The data becomes available in Amazon Redshift. You control the encryption of your data when you create the integration source, when you create the zero-ETL integration, and when you create the Amazon Redshift data warehouse. The integration monitors

the health of the data pipeline and recovers from issues when possible. You can create integrations from sources of the same type into a single Amazon Redshift data warehouse to derive holistic insights across multiple applications.

With the data in Amazon Redshift, you can use analytics that Amazon Redshift provides. For example, built-in machine learning (ML), materialized views, data sharing, and direct access to multiple data stores and data lakes. For data engineers, zero-ETL integration provides access to time-sensitive data that otherwise can get delayed by intermittent errors in complex data pipelines. You can run analytical queries and ML models on transactional data to derive timely insights for time-sensitive events and business decisions.

You can create an Amazon Redshift event notification subscription so you can be notified when an event occurs for a given zero-ETL integration. To view the list of integration-related event notifications, see Zero-ETL integration event notifications with Amazon EventBridge. The simplest way to create a subscription is with the Amazon SNS console. For information on creating an Amazon SNS topic and subscribing to it, see Getting started with Amazon SNS in the *Amazon Simple Notification Service Developer Guide*.

As you get started with zero-ETL integrations, consider the following concepts:

- A source database is the database from where data is replicated into Amazon Redshift.
- A target data warehouse is the Amazon Redshift provisioned cluster or Redshift Serverless workgroup where data is replicated to.
- A destination database is the database that you create from a zero-ETL integration in the target data warehouse.

For information about system tables and views you can use to monitor your zero-ETL integrations, see Monitoring zero-ETL integrations with Amazon Redshift system views.

For a list of AWS Regions that each source for zero-ETL integrations supports, see Supported Regions for zero-ETL integrations.

For pricing information for zero-ETL integrations, see the appropriate pricing page:

- Amazon Redshift pricing
- Amazon Aurora pricing
- Amazon RDS pricing
- Amazon DynamoDB pricing

- AWS Glue pricing

For more information about zero-ETL integration sources, see the following topics:

- For Aurora zero-ETL integrations, see Benefits, Key concepts, Limitations, Quotas, and Supported Regions of zero-ETL integrations in the *Amazon Aurora User Guide*.
- For RDS zero-ETL integrations, see Benefits, Key concepts, Limitations, Quotas, and Supported Regions of zero-ETL integrations in the *Amazon RDS User Guide*.
- For DynamoDB zero-ETL integrations, see DynamoDB zero-ETL integration with Amazon Redshift in the *Amazon DynamoDB Developer Guide*.
- For zero-ETL integrations with applicatons, see Zero-ETL integrations in the *AWS Glue Developer Guide*.

Topics

- Considerations when using zero-ETL integrations with Amazon Redshift
- Getting started with zero-ETL integrations
- Creating destination databases in Amazon Redshift
- Querying replicated data in Amazon Redshift
- Viewing zero-ETL integrations
- History mode
- Sharing your data in Amazon Redshift
- Monitoring zero-ETL integrations
- Metrics for zero-ETL integrations
- Modify a zero-ETL integration for DynamoDB
- Delete a zero-ETL integration for DynamoDB
- Supported Regions for zero-ETL integrations
- Troubleshooting zero-ETL integrations

Considerations when using zero-ETL integrations with Amazon Redshift

The following considerations apply to zero-ETL integrations with Amazon Redshift.

- Your target Amazon Redshift data warehouse must meet the following prerequisites:
  - Running Amazon Redshift Serverless or a provisioned cluster of an RG or RA3 node type.
  - Encrypted (if using a provisioned cluster).
  - Has case sensitivity enabled.
- If you delete a source that is an authorized integration source for an Amazon Redshift data warehouse, all associated integrations will go into the `FAILED` state. Any previously replicated data remains in your Amazon Redshift database and can be queried.
- The destination database is read-only. You can't create tables, views, or materialized views in the destination database. However, you can use materialized views on other tables in the target data warehouse.
- Materialized views are supported when used in cross-database queries. For information about creating materialized views with data replicated through zero-ETL integrations, see Querying replicated data with materialized views.
- By default, you can query tables only in the target data warehouse that are in the `Synced` state. To query tables in another state, set the database parameter `QUERY_ALL_STATES` to `TRUE`. For information about setting `QUERY_ALL_STATES`, see CREATE DATABASE and ALTER DATABASE in the *Amazon Redshift Database Developer Guide*. For more information about the state of your database, see SVV_INTEGRATION_TABLE_STATE in the *Amazon Redshift Database Developer Guide*.
- Amazon Redshift accepts only UTF-8 characters, so it might not honor the collation defined in your source. The sorting and comparison rules might be different, which can ultimately change the query results.
- Zero-ETL integrations is limited to 50 per Amazon Redshift data warehouse target.
- Tables in the integration source must have a primary key. Otherwise, your tables can't be replicated to the target data warehouse in Amazon Redshift.

  For information about how to add a primary key to Amazon Aurora PostgreSQL, see Handle tables without primary keys while creating Amazon Aurora PostgreSQL zero-ETL integrations with Amazon Redshift in the *AWS Database Blog*. For information about how to add a primary key to Amazon Aurora MySQL or RDS for MySQL, see Handle tables without primary keys while creating Amazon Aurora MySQL or Amazon RDS for MySQL zero-ETL integrations with Amazon Redshift in the *AWS Database Blog*.
- You can use data filtering for Aurora zero-ETL integrations to define the scope of replication from the source Aurora DB cluster to the target Amazon Redshift data warehouse. Rather than replicating all data to the target, you can define one or more filters that selectively include or

exclude certain tables from being replicated. For more information, see Data filtering for Aurora zero-ETL integrations with Amazon Redshift in the *Amazon Aurora User Guide*.

- For Aurora PostgreSQL zero-ETL integrations with Amazon Redshift, Amazon Redshift supports a maximum of 100 databases from Aurora PostgreSQL. Each database replicates from source to target independently.
- Zero-ETL integration does not support transformations while replicating the data from transactional data stores to Amazon Redshift. Data is replicated as-is from the source data base. However, you can apply transformations on the replicated data in Amazon Redshift.
- Zero-ETL integration runs in Amazon Redshift using parallel connections. It runs using the credentials of the user who created the database from the integration.
- You can set the `REFRESH_INTERVAL` for a zero-ETL integration to control the frequency of data replication into Amazon Redshift. For more information, see CREATE DATABASE and ALTER DATABASE in the *Amazon Redshift Database Developer Guide*.
- After you create an Amazon Redshift database from a zero-ETL integration with Amazon DynamoDB, the database state should change from **Creating** to **Active**. This starts the replication of data in the source DynamoDB tables to the target Redshift tables, which are created under the public schema of the destination database (`ddb_rs_customerprofiles_zetl_db`).

Considerations when using history mode on the target

The following considerations apply when using history mode on the target database. For more information, see History mode.

- When you drop a table on a source, the table on the target is not dropped, but is changed to `DroppedSource` state. You can drop or rename the table from the Amazon Redshift database.
- When you truncate a table on a source, deletes are run on the target table. For example, if all records are truncated on the source, corresponding records on the target column `_record_is_active` are changed to `false`.
- When you run TRUNCATE table SQL on the target table, active history rows are marked inactive with a corresponding timestamp.
- When a row in a table is set to inactive, it can be deleted only after zero-ETL persists its inactive state in the table. Queries show the row as inactive immediately. However, zero-ETL takes longer to persist that state in the background, usually within an hour but can take longer for a table with infrequent changes. Additional changes to the source table can cause the inactive state to be persisted earlier. A DELETE that runs before the inactive state is persisted completes

successfully, but leaves the inactive row in place. If you see the inactive rows after a successful DELETE command, run the DELETE command again later, for example on a schedule. To delete inactive rows, connect to your zero-ETL database with query editor v2 or another SQL client.

- You can only delete inactive rows from a table with history mode on. For example, a SQL command similar to the following only deletes inactive rows.

```
delete from schema.user_table where _record_delete_time <= '2024-09-10 12:34:56'
```

  This is equivalent to a SQL command like the following.

```
delete from schema.user_table where _record_delete_time <= '2024-09-10 12:34:56' and
 _record_is_active = False
```

- When turning history mode off for a table, all historical data is saved to table named with `<schema>.<table-name>_historical_<timestamp>` while the original table named `<schema>.<table-name>` is refreshed.
- When a table with history mode on is excluded from replication using a table filter, all rows are set as inactive and it is changed to `DroppedSource` state. For more information about table filters, see Data filtering for Aurora zero-ETL integrations with Amazon Redshift in the *Amazon Aurora User Guide*.
- History mode can only be switched to `true` or `false` for tables in `Synced` state.
- Materialized views for tables with history mode on are created as full recompute.

Considerations when the zero-ETL integration source is Aurora or Amazon RDS

The following considerations apply to Aurora and Amazon RDS zero-ETL integrations with Amazon Redshift.

- You can use data filtering for Aurora and RDS for MySQL zero-ETL integrations to define the scope of replication from the source DB cluster to the target Amazon Redshift data warehouse. Rather than replicating all data to the target, you can define one or more filters that selectively include or exclude certain tables from being replicated. For more information, see Data filtering for Aurora zero-ETL integrations with Amazon Redshift in the *Amazon Aurora User Guide*.
- Tables in the integration source must have a primary key. Otherwise, your tables can't be replicated to the target data warehouse in Amazon Redshift.

For information about how to add a primary key to Amazon Aurora PostgreSQL, see Handle tables without primary keys while creating Amazon Aurora PostgreSQL zero-ETL integrations with Amazon Redshift in the *AWS Database Blog*. For information about how to add a primary key to Amazon Aurora MySQL or RDS for MySQL, see Handle tables without primary keys while creating Amazon Aurora MySQL or Amazon RDS for MySQL zero-ETL integrations with Amazon Redshift in the *AWS Database Blog*.

- The maximum length of an Amazon Redshift VARCHAR data type is 65,535 bytes. When the content from the source does not fit into this limit, replication does not proceed and the table is put into a failed state. You can set the database parameter `TRUNCATECOLUMNS` to `TRUE` to truncate content to fit in the column. For information about setting `TRUNCATECOLUMNS`, see CREATE DATABASE and ALTER DATABASE in the *Amazon Redshift Database Developer Guide*.

  For more information about data type differences between zero-ETL integration sources and Amazon Redshift databases, see Data type differences between Aurora and Amazon Redshift in the *Amazon Aurora User Guide*.

For Aurora sources, also see Limitations in the *Amazon Aurora User Guide*.

For Amazon RDS sources, also see Limitations in the *Amazon RDS User Guide*.

Considerations when the zero-ETL integration source is DynamoDB

The following considerations apply to DynamoDB zero-ETL integrations with Amazon Redshift.

- Table names from DynamoDB greater than 127 characters are not supported.
- The data from a DynamoDB zero-ETL integration maps to a SUPER data type column in Amazon Redshift.
- Column names for the partition key or sort key greater than 127 characters are not supported.
- A zero-ETL integration from DynamoDB can map to only one Amazon Redshift database.
- For partition and sort keys, the precision and scale maximum is (38,18). Numeric data types on DynamoDB support a maximum precision up to 38. Amazon Redshift also supports a maximum precision of 38, but the default decimal precision/scale on Amazon Redshift is (38,10). That means values scale values can be truncated.
- For a successful zero-ETL integration, an individual attribute (consisting of name+value) in a DynamoDB item, must not be larger than 64 KB.

- On activation, the zero-ETL integration exports the full DynamoDB table to populate the Amazon Redshift database. The time it takes for this initial process to complete depends on the DynamoDB table size. The zero-ETL integration then incrementally replicates updates from DynamoDB to Amazon Redshift using DynamoDB incremental exports. This means the replicated DynamoDB data in Amazon Redshift is kept up-to-date automatically.

  Currently, the minimum latency for DynamoDB zero-ETL integration is 15 minutes. You can increase it further by setting a non-zero `REFRESH_INTERVAL` for a zero-ETL integration. For more information, see CREATE DATABASE and ALTER DATABASE in the *Amazon Redshift Database Developer Guide*.

For Amazon DynamoDB sources, also see Prerequisites and limitations in the *Amazon DynamoDB Developer Guide*.

Considerations when the zero-ETL integration source is applications, such as, Salesforce, SAP, ServiceNow, and Zendesk

The following considerations apply to source is applications, such as, Salesforce, SAP, ServiceNow, and Zendesk with Amazon Redshift.

- Table names and column names from application sources greater than 127 characters are not supported.
- The maximum length of an Amazon Redshift VARCHAR data type is 65,535 bytes. When the content from the source does not fit into this limit, replication does not proceed and the table is put into a failed state. You can set the database parameter `TRUNCATECOLUMNS` to `TRUE` to truncate content to fit in the column. For information about setting `TRUNCATECOLUMNS` see CREATE DATABASE and ALTER DATABASE in the *Amazon Redshift Database Developer Guide*.

  For more information about data type differences between zero-ETL integration application sources and Amazon Redshift databases, see Zero-ETL integrations in the *AWS Glue Developer Guide*.
- The minimum latency for a zero-ETL integration with applications is 1 hour. You can increase it further by setting a non-zero `REFRESH_INTERVAL` for a zero-ETL integration. For more information, see CREATE DATABASE and ALTER DATABASE in the *Amazon Redshift Database Developer Guide*.

For sources of zero-ETL integrations with applications, also see Zero-ETL integrations in the *AWS Glue Developer Guide*.

Considerations when restoring a zero-ETL integration target

When you restore an Amazon Redshift Serverless namespace from a snapshot or recovery point to the same serverless namespace, zero-ETL integrations associated with the namespace are maintained automatically. A full resync is initiated after the restore to ensure data in the target Amazon Redshift database is consistent with the source. The following considerations apply:

- Restoring a snapshot to a different namespace does not maintain integrations.
- After the restore, Amazon Redshift performs a full resync for all maintained zero-ETL integrations. During the resync, data ingestion performance might be temporarily reduced.
- If you use history mode on the target database, record versions from the time the snapshot was taken until the full resync completes are not captured. After the resync, history mode resumes normal operation. This behavior is the same as any other full resync. For more information, see History mode.
- If a zero-ETL integration was created after the snapshot was taken, the integration enters the `NEEDS_ATTENTION` state after the restore because the corresponding database does not exist in the snapshot. To resolve this, you can restore from a more recent snapshot that includes the integration, or delete the integration.
- If a zero-ETL integration was deleted after the snapshot was taken, the integration is not restored. The database from the snapshot remains and can be deleted.
- If table-level filters were changed after the snapshot was taken, the restored target uses the current filters from the integration, not the filters at the time of the snapshot.
- To opt out of maintaining integrations during a restore, uncheck the **Maintain Integrations** box on the restore page in the AWS Management Console, or if you are using the AWS CLI, set the `--no-maintain-integration` parameter when calling the restore-from-snapshot or restore-from-recovery-point API operation. When you opt out, integrations enter the `FAILED` state after the restore. You can delete the integrations and recreate them.
- This feature applies to Amazon Redshift Serverless only when restored to the same serverless namespace. Snapshot restores on provisioned clusters do not maintain zero-ETL integrations.

Considerations when resizing a zero-ETL integration target

When you resize an Amazon Redshift provisioned cluster that is the target of a zero-ETL integration, the resize can cause the integration tables to resynchronize. Whether resynchronization occurs depends on the type of resize:

- An elastic resize that changes only the number of nodes (an in-place resize) does not affect zero-ETL integrations. Tables remain synchronized.
- An elastic resize that changes the node type causes all tables in zero-ETL integrations on the cluster to resynchronize. Any classic resize also triggers resynchronization. This happens because these resize operations temporarily change the distribution style of tables while the service redistributes data onto the new cluster configuration. For more information, see Resizing a cluster.

The following considerations apply when a resize triggers resynchronization:

- The integration remains active, and resynchronization starts automatically after the resize completes. You don't need to take any action.
- While a table is resynchronizing, you can't query it in Amazon Redshift. To keep tables queryable during resynchronization, set the `QUERY_ALL_STATES` parameter to `TRUE` on the destination database before you start the resize. Data returned during resynchronization might be stale until the resynchronization completes. For more information, see CREATE DATABASE and ALTER DATABASE in the *Amazon Redshift Database Developer Guide*.
- Resynchronization can take 20–25 minutes or more, depending on the size of the source database.
- You can monitor the state of integration tables using the SVV_INTEGRATION_TABLE_STATE system view. Tables show the `ResyncRequired` or `ResyncInitiated` state until resynchronization completes and they return to `Synced`.

Getting started with zero-ETL integrations

This set of tasks walks you through setting up your first zero-ETL integration. First, you configure your integration source and set it up with the required parameters and permissions. Then, you continue to the rest of the initial setup from the Amazon Redshift console or AWS CLI. The console provides a **Fix it for me** option to correct some configuration issues.

Topics

- Create and configure a target Amazon Redshift data warehouse
- Turn on case sensitivity for your data warehouse
- Configure authorization for your Amazon Redshift data warehouse
- Create a zero-ETL integration

Create and configure a target Amazon Redshift data warehouse

In this step, you create and configure a target Amazon Redshift data warehouse, such as a Redshift Serverless workgroup or a provisioned cluster. If you already have a Amazon Redshift data warehouse configured for use with zero-ETL integrations, you can skip this step.

Your target data warehouse must have the following characteristics:

- Running Amazon Redshift Serverless or a provisioned cluster of an RG or RA3 node type.
- Has case sensitivity (`enable_case_sensitive_identifier`) turned on. For more information, see Turn on case sensitivity for your data warehouse.
- Encrypted, if your target data warehouse is an Amazon Redshift provisioned cluster. For more information, see Amazon Redshift database encryption.
- Created in the same AWS Region as the integration source.

To create your target data warehouse for your zero-ETL integrations, see one of the following topics depending on your deployment type:

- To create an Amazon Redshift provisioned cluster, see Creating a cluster.
- To create an Amazon Redshift Serverless workgroup with a namespace, see Creating a workgroup with a namespace.

When you create a provisioned cluster, Amazon Redshift also creates a default parameter group. You can't edit the default parameter group. However, you can create a custom parameter group before creating a new cluster and then associate it with the cluster. Or, you can edit the parameter group that will be associated with the created cluster. You must also turn on case sensitivity for the parameter group either when creating the custom parameter group or when editing a current one to use zero-ETL integrations.

To create a custom parameter group using the Amazon Redshift console or the AWS CLI, see Creating a parameter group.

Turn on case sensitivity for your data warehouse

You can attach a parameter group and enable case sensitivity for a provisioned cluster during creation. However, you can update a serverless workgroup through the AWS Command Line Interface (AWS CLI) only after it's been created. This is required to support the case sensitivity of source tables and columns. The `enable_case_sensitive_identifier` is a configuration value that determines whether name identifiers of databases, tables, and columns are case sensitive. This parameter must be turned on to create zero-ETL integrations in the data warehouse. For more information, see enable_case_sensitive_identifier.

For Amazon Redshift Serverless – Turn on case sensitivity for Amazon Redshift Serverless using the AWS CLI. Note that you can turn on case sensitivity for Amazon Redshift Serverless only from the AWS CLI.

For Amazon Redshift provisioned clusters, enable case sensitivity for your target cluster using one of the following topics:

- Turn on case sensitivity for Amazon Redshift provisioned clusters using the Amazon Redshift console
- Turn on case sensitivity for Amazon Redshift provisioned clusters using the AWS CLI

Turn on case sensitivity for Amazon Redshift Serverless using the AWS CLI

Run the following AWS CLI command to turn on case sensitivity for your workgroup.

```
aws redshift-serverless update-workgroup \
        --workgroup-name target-workgroup \
        --config-parameters
 parameterKey=enable_case_sensitive_identifier,parameterValue=true
```

Wait for the workgroup status to be `Active` before proceeding to the next step.

Turn on case sensitivity for Amazon Redshift provisioned clusters using the Amazon Redshift console

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. In the left navigation pane, choose **Provisioned clusters dashboard**.
3. Choose the provisioned cluster that you want to replicate data into.
4. In the left navigation pane, choose **Configurations** > **Workload management**.
5. On the workload management page, choose the parameter group.
6. Choose the **Parameters** tab.
7. Choose **Edit parameters**, then change **enable_case_sensitive_identifier** to **true**.
8. Then, choose **Save**.

Turn on case sensitivity for Amazon Redshift provisioned clusters using the AWS CLI

1. Because you can't edit the default parameter group, from your terminal program, run the following AWS CLI command to create a custom parameter group. Later, you will associate it with the provisioned cluster.

```
aws redshift create-cluster-parameter-group \
    --parameter-group-name zero-etl-params \
    --parameter-group-family redshift-2.0 \
    --description "Param group for zero-ETL integrations"
```

2. Run the following AWS CLI command to turn on case sensitivity for the parameter group.

```
aws redshift modify-cluster-parameter-group \
    --parameter-group-name zero-etl-params \
    --parameters ParameterName=enable_case_sensitive_identifier,ParameterValue=true
```

3. Run the following command to associate the parameter group with the cluster.

```
aws redshift modify-cluster \
    --cluster-identifier target-cluster \
    --cluster-parameter-group-name zero-etl-params
```

4. Wait for the provisioned cluster to be available. You can check the status of the cluster by using the `describe-cluster` command. Then, run the following command to reboot the cluster.

```
aws redshift reboot-cluster \
    --cluster-identifier target-cluster
```

Configure authorization for your Amazon Redshift data warehouse

To replicate data from your integration source into your Amazon Redshift data warehouse, you must initially add the following two entities:

- *Authorized principal* – identifies the user or role that can create zero-ETL integrations into the data warehouse.
- *Authorized integration source* – identifies the source database that can update the data warehouse.

You can configure authorized principals and authorized integration sources from the **Resource Policy** tab on the Amazon Redshift console or using the Amazon Redshift `PutResourcePolicy` API operation.

Add authorized principals

To create a zero-ETL integration into your Redshift Serverless workgroup or provisioned cluster, authorize access to the associated namespace or provisioned cluster.

You can skip this step if both of the following conditions are true:

- The AWS account that owns the Redshift Serverless workgroup or provisioned cluster also owns the source database.
- That principal is associated with an identity-based IAM policy with permissions to create zero-ETL integrations into this Redshift Serverless namespace or provisioned cluster.

Add authorized principals to an Amazon Redshift Serverless namespace

1. In the Amazon Redshift console, in the left navigation pane, choose **Redshift Serverless**.

2. Choose **Namespace configuration**, then choose your namespace, and go to the **Resource Policy** tab.
3. Choose **Add authorized principals**.
4. For each authorized principal that you want to add, enter into the namespace either the ARN of the AWS user or role, or the ID of the AWS account that you want to grant access to create zero-ETL integrations. An account ID is stored as an ARN.
5. Choose **Save changes**.

Add authorized principals to an Amazon Redshift provisioned cluster

1. In the Amazon Redshift console, in the left navigation pane, choose **Provisioned clusters dashboard**.
2. Choose **Clusters**, then choose the cluster, and go to the **Resource Policy** tab.
3. Choose **Add authorized principals**.
4. For each authorized principal that you want to add, enter into the cluster either the ARN of the AWS user or role, or the ID of the AWS account that you want to grant access to create zero-ETL integrations. An account ID is stored as an ARN.
5. Choose **Save changes**.

Add authorized integration sources

To allow your source to update your Amazon Redshift data warehouse, you must add it as an authorized integration source to the namespace.

Add an authorized integration source to an Amazon Redshift Serverless namespace

1. In the Amazon Redshift console, go to **Serverless dashboard**.
2. Choose the name of the namespace.
3. Go to the **Resource Policy** tab.
4. Choose **Add authorized integration source**.
5. Specify the ARN of the source for the zero-ETL integration.

> **Note**
>
> Removing an authorized integration source stops data from replicating into the namespace. This action deactivates all zero-ETL integrations from that source into this namespace.

Add an authorized integration source to an Amazon Redshift provisioned cluster

1. In the Amazon Redshift console, go to **Provisioned clusters dashboard**.
2. Choose the name of the provisioned cluster.
3. Go to the **Resource Policy** tab.
4. Choose **Add authorized integration source**.
5. Specify the ARN of the source that's the data source for the zero-ETL integration.

> **Note**
>
> Removing an authorized integration source stops data from replicating into the provisioned cluster. This action deactivates all zero-ETL integrations from that source into this Amazon Redshift provisioned cluster.

Configure authorization using the Amazon Redshift API

You can use the Amazon Redshift API operations to configure resource policies that work with zero-ETL integrations.

To control the source that can create an inbound integration into the namespace, create a resource policy and attach it to the namespace. With the resource policy, you can specify the source that has access to the integration. The resource policy is attached to the namespace of your target data warehouse to allow the source to create an inbound integration to replicate live data from the source into Amazon Redshift.

The following is a sample resource policy.

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift.amazonaws.com"
      },
      "Action": "redshift:AuthorizeInboundIntegration",
      "Resource": "arn:aws:redshift-serverless:us-east-1:123456789012:namespace/
cc4ffe56-ad2c-4fd1-a5a2-f29124a56433",
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:rds:us-east-1:111122223333:cluster:foo"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "redshift:CreateInboundIntegration",
      "Resource": "arn:aws:redshift-serverless:us-east-1:123456789012:namespace/
cc4ffe56-ad2c-4fd1-a5a2-f29124a56433"
    }
  ]
}
```

The following summarizes the Amazon Redshift API operations applicable to configuring resource policies for integrations:

- Use the PutResourcePolicy API operation to persist the resource policy. When you provide another resource policy, the previous resource policy on the resource is replaced. Use the previous example resource policy, which grants permissions for the following actions:
  - `CreateInboundIntegration` – Allows the source principal to create an inbound integration for data to be replicated from the source into the target data warehouse.
  - `AuthorizeInboundIntegration` – Allows Amazon Redshift to continuously validate that the target data warehouse can receive data replicated from the source ARN.
- Use the GetResourcePolicy API operation is to view existing resource policies.
- Use the DeleteResourcePolicy API operation to remove a resource policy from the resource.

To update a resource policy, you can also use the put-resource-policy AWS CLI command. For example, to put a resource policy on your Amazon Redshift namespace ARN for a DynamoDB source, run a AWS CLI command similar to the following.

```
aws redshift put-resource-policy \
--policy file://rs-rp.json \
--resource-arn "arn:aws:redshift-serverless:us-east-1:123456789012:namespace/cc4ffe56-ad2c-4fd1-a5a2-f29124a56433"
```

Where `rs-rp.json` contains:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "redshift.amazonaws.com"
            },
            "Action": "redshift:AuthorizeInboundIntegration",
            "Resource": "arn:aws:redshift-serverless:us-east-1:123456789012:namespace/cc4ffe56-ad2c-4fd1-a5a2-f29124a56433",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": "arn:aws:dynamodb:us-east-1:123456789012:table/test_ddb"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:root"
            },
            "Action": "redshift:CreateInboundIntegration",
            "Resource": "arn:aws:redshift-serverless:us-east-1:123456789012:namespace/cc4ffe56-ad2c-4fd1-a5a2-f29124a56433"
        }
    ]
```

```
}
```

Create a zero-ETL integration

First, you create a zero-ETL integration to replicate your source data to Amazon Redshift.

The source of your data determines which type of zero-ETL integration to create.

Topics

- Create a zero-ETL integration for Aurora
- Create a zero-ETL integration for Amazon RDS
- Create a zero-ETL integration for DynamoDB
- Create a zero-ETL integration with applications

Create a zero-ETL integration for Aurora

In this step, you create an Aurora zero-ETL integration with Amazon Redshift.

To create an Aurora zero-ETL integration with Amazon Redshift

1. From the Amazon RDS console, create a custom DB cluster parameter group as described in the *Amazon Aurora User Guide*.
2. From the Amazon RDS console, create a source Amazon Aurora DB cluster as described in the *Amazon Aurora User Guide*.
3. From the Amazon Redshift console: Create and configure a target Amazon Redshift data warehouse.
   - From the AWS CLI or Amazon Redshift console: Turn on case sensitivity for your data warehouse.
   - From the Amazon Redshift console: Configure authorization for your Amazon Redshift data warehouse.
4. From the Amazon RDS console, create a zero-ETL integration as described in the *Amazon Aurora User Guide*.
5. From the Amazon Redshift console or the query editor v2, create an Amazon Redshift database from your integration.

Then, query and create materialized views with replicated data.

For detailed information to create Aurora zero-ETL integrations, see Creating Amazon Aurora zero-ETL integrations with Amazon Redshift in the *Amazon Aurora User Guide*.

Create a zero-ETL integration for Amazon RDS

In this step, you create an Amazon RDS zero-ETL integration with Amazon Redshift. Redshift supports integrations with RDS for MySQL, RDS for PostgreSQL, and RDS for Oracle.

To create an Amazon RDS zero-ETL integration with Amazon Redshift

1. From the Amazon RDS console, create a custom DB parameter group as described in the *Amazon RDS User Guide*.
2. From the Amazon RDS console, create a source Amazon RDS instance as described in the *Amazon RDS User Guide*.
3. From the Amazon Redshift console: Create and configure a target Amazon Redshift data warehouse.
   - From the AWS CLI or Amazon Redshift console: Turn on case sensitivity for your data warehouse.
   - From the Amazon Redshift console: Configure authorization for your Amazon Redshift data warehouse.
4. From the Amazon RDS console, create a zero-ETL integration as described in the *Amazon RDS User Guide*.
5. From the Amazon Redshift console or the query editor v2, create an Amazon Redshift database from your integration.

   Then, query and create materialized views with replicated data.

The Amazon RDS console offers a step-by-step integration creation flow, in which you specify the source database and the target Amazon Redshift data warehouse. If issues occur, then you can choose to have Amazon RDS fix the issues for you instead of manually fixing them on either the Amazon RDS or Amazon Redshift console.

For detailed instructions to create RDS zero-ETL integrations, see Creating Amazon RDS zero-ETL integrations with Amazon Redshift in the *Amazon RDS User Guide*.

For detailed instructions to specifically create an Amazon RDS for Oracle zero-ETL integration, see Setting up a zero-ETL integration in the *Oracle Database@AWS User Guide*.

Create a zero-ETL integration for DynamoDB

Before creating a zero-ETL integration, review the considerations and requirements outlined in Considerations when using zero-ETL integrations with Amazon Redshift. Follow this general flow to create a zero-ETL integration from DynamoDB to Amazon Redshift

To replicate DynamoDB data to Amazon Redshift with zero-ETL integration

1. Confirm your sign in credentials allow permissions to work with zero-ETL integrations with Amazon Redshift and DynamoDB. See IAM policy to work with DynamoDB zero-ETL integrations for an example IAM policy.
2. From the DynamoDB console, configure your DynamoDB table to have point-in-time recovery (PITR), resource policies, identity-based policies, and encryption key permissions as described in the *Amazon DynamoDB Developer Guide*.
3. From the Amazon Redshift console: Create and configure a target Amazon Redshift data warehouse.
   - From the AWS CLI or Amazon Redshift console: Turn on case sensitivity for your data warehouse.
   - From the Amazon Redshift console: Configure authorization for your Amazon Redshift data warehouse.
4. From the Amazon Redshift console, create the zero-ETL integration integration as described later in this topic.
5. From the Amazon Redshift console, create the destination database in your Amazon Redshift data warehouse. For more information, see Creating destination databases in Amazon Redshift.
6. From the Amazon Redshift console, query your replicated data in the Amazon Redshift data warehouse. For more information, see Querying replicated data in Amazon Redshift.

In this step, you create an Amazon DynamoDB zero-ETL integration with Amazon Redshift.

Amazon Redshift console

To create an Amazon DynamoDB zero-ETL integration with Amazon Redshift using the Amazon Redshift console

1. From the Amazon Redshift console, choose **Zero-ETL integrations**. On the pane with the list of zero-ETL integrations, choose **Create zero-ETL integration**, **Create DynamoDB integration**.
2. On the pages to create an integration, enter information about the integration as follows:

   - Enter an **Integration name** – Which is a unique name that can be used to reference your integration.
   - Enter a **Description** – That describes the data that is to be replicated from source to target.
   - Choose the DynamoDB **Source table** – One DynamoDB table can be chosen. Point-in-time recovery (PITR) must be enabled on the table. Only tables with a table size up to 100 tebibytes (TiB) are shown. The source DynamoDB table must be encrypted. The source must also have a resource policy with authorized principals and integration sources. If these the policy is not correct, you are presented with the option **Fix it for me**.
   - Choose the target **Amazon Redshift data warehouse** – The data warehouse can be an Amazon Redshift provisioned cluster or Redshift Serverless workgroup. If your target Amazon Redshift is in the same account, you are able to select the target. If the target is in a different account, you specify the **Redshift data warehouse ARN**. The target must have a resource policy with authorized principals and integration source and the `enable_case_sensitive_identifier` parameter set to true. If you do not have the correct resource policies on the target and your target is in the same account, you can select the **Fix it for me** option to automatically apply the resource policies during the create integration process. If your target is in a different AWS account, you need to apply the resource policy on the Amazon Redshift warehouse manually. If your target Amazon Redshift data warehouse does not have the correct parameter group option `enable_case_sensitive_identifier` configured as `true`, you can select the **Fix it for me** option to automatically update this parameter group and reboot the warehouse during the create integration process.
   - Enter up to 50 tag **Keys** and with an optional **Value** – To provide additional metadata about the integration. For more information, see Tag resources in Amazon Redshift.

- Choose **Encryption** options – To encrypt the integration. For more information, see Encrypting DynamoDB integrations with a customer managed key.

  When you encrypt the integration, you can also add **Additional encryption contexts**. For more information, see Encryption context.

3. A review page is shown where you can choose **Create DynamoDB integration**.
4. A progress page is shown where you can view the progress of the various tasks to create the zero-ETL integration.
5. After the integration is created and active, on the details page of the integration, choose **Connect to database**. When your Amazon Redshift data warehouse was first created, a database was also created. You need to connect to any database in your target data warehouse to create another database for the integration. In the **Connect to database** page, determine whether you can use a recent connection and choose an **Authentication** method. Depending on your authentication method, enter information to connect to an existing database in your target. This authentication information can include the existing **Database name** (typically, `dev`) and the **Database user** specified when the database was created with the Amazon Redshift data warehouse.
6. After you are connected to a database, choose **Create database from integration** to create the database that receives the data from the source. When you create the database you provide the **Integration ID**, **Data warehouse name**, and **Database name**.
7. After the integration status and destination database is `Active`, data begins to replicate from your DynamoDB table to the target table. As you add data to the source it replicates automatically to the target Amazon Redshift data warehouse.

AWS CLI

To create an Amazon DynamoDB zero-ETL integration with Amazon Redshift using the AWS CLI, use the `create-integration` command with the following options:

- `integration-name` – Specify a name for the integration.
- `source-arn` – Specify the ARN of the DynamoDB source.
- `target-arn` – Specify the namespace ARN of the Amazon Redshift provisioned cluster or Redshift Serverless workgroup target.

The following example creates an integration by providing the integration name, source ARN, and target ARN. The integration is not encrypted.

```
aws redshift create-integration \
--integration-name ddb-integration \
--source-arn arn:aws:dynamodb:us-east-1:123456789012:table/books \
--target-arn arn:aws:redshift:us-east-1:123456789012:namespace:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222

{
    "Status": "creating",
    "IntegrationArn": "arn:aws:redshift:us-east-1:123456789012:integration:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "Errors": [],
    "ResponseMetadata": {
        "RetryAttempts": 0,
        "HTTPStatusCode": 200,
        "RequestId": "132cbe27-fd10-4f0a-aacb-b68f10bb2bfb",
        "HTTPHeaders": {
            "x-amzn-requestid": "132cbe27-fd10-4f0a-aacb-b68f10bb2bfb",
            "date": "Sat, 24 Aug 2024 05:44:08 GMT",
            "content-length": "934",
            "content-type": "text/xml"
        }
    },
    "Tags": [],
    "CreateTime": "2024-08-24T05:44:08.573Z",
    "KMSKeyId": "arn:aws:kms:us-east-1:123456789012:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "AdditionalEncryptionContext": {},
    "TargetArn": "arn:aws:redshift:us-east-1:123456789012:namespace:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "IntegrationName": "ddb-integration",
    "SourceArn": "arn:aws:dynamodb:us-east-1:123456789012:table/books"
}
```

The following example creates an integration using a customer managed key for encryption. Before creating the integration:

- Create a customer managed key (called “CMCMK” in the example) in the same account (called “AccountA” in the example) at the source DynamoDB table.

- Ensure that the user/role (called “RoleA” in the example) is being used to create the integration has `kms:CreateGrant` and `kms:DescribeKey` permissions on this KMS key.
- Add the following to the key policy.

```
{
    "Sid": "Enable RoleA to create grants with key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "RoleA-ARN"
    },
    "Action": "kms:CreateGrant",
    "Resource": "*",
    "Condition": {
        // Add "StringEquals" condition if you plan to provide additional encryption
 context
        // for the zero-ETL integration. Ensure that the key-value pairs added here
 match
        // the key-value pair you plan to use while creating the integration.
        // Remove this if you don't plan to use additional encryption context
        "StringEquals": {
            "kms:EncryptionContext:context-key1": "context-value1"
        },
        "ForAllValues:StringEquals": {
            "kms:GrantOperations": [
                "Decrypt",
                "GenerateDataKey",
                "CreateGrant"
            ]
        }
    }
},
{
    "Sid": "Enable RoleA to describe key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "RoleA-ARN"
    },
    "Action": "kms:DescribeKey",
    "Resource": "*"
},
{
    "Sid": "Allow use by RS SP",
```

```
    "Effect": "Allow",
    "Principal": {
        "Service": "redshift.amazonaws.com"
           },
    "Action": "kms:CreateGrant",
    "Resource": "*"
}
```

```
aws redshift create-integration \
--integration-name ddb-integration \
--source-arn arn:aws:dynamodb:us-east-1:123456789012:table/books \
--target-arn arn:aws:redshift:us-east-1:123456789012:namespace:a1b2c3d4-5678-90ab-
cdef-EXAMPLE22222 \
--kms-key-id arn:aws:kms:us-east-1:123456789012:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE33333 \
--additional-encryption-context key33=value33  // This matches the condition in the
 key policy.
          {
    "IntegrationArn": "arn:aws:redshift:us-
east-1:123456789012:integration:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "IntegrationName": "ddb-integration",
    "SourceArn": "arn:aws:dynamodb:us-east-1:123456789012:table/books",
    "SourceType": "dynamodb",
    "TargetArn": "arn:aws:redshift:us-
east-1:123456789012:namespace:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "Status": "creating",
    "Errors": [],
    "CreateTime": "2024-10-02T18:29:26.710Z",
    "KMSKeyId": "arn:aws:kms:us-east-1:123456789012:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE33333",
    "AdditionalEncryptionContext": {
        "key33": "value33"
    },
    "Tags": []
}
```

IAM policy to work with DynamoDB zero-ETL integrations

When creating zero-ETL integrations, your sign in credentials must have permission to on both DynamoDB and Amazon Redshift actions and also on the resources involved as sources and targets of the integration. Following is a example that demonstrates the minimum permissions required.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:ListTables"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetResourcePolicy",
                "dynamodb:PutResourcePolicy",
                "dynamodb:UpdateContinuousBackups"
            ],
            "Resource": [
            "arn:aws:dynamodb:us-east-1:111122223333:table/my-ddb-table"
            ]
        },
        {
            "Sid": "AllowRedshiftDescribeIntegration",
            "Effect": "Allow",
            "Action": [
                "redshift:DescribeIntegrations"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowRedshiftCreateIntegration",
            "Effect": "Allow",
            "Action": "redshift:CreateIntegration",
            "Resource": "arn:aws:redshift:us-east-1:111122223333:integration:*"
        },
        {
            "Sid": "AllowRedshiftModifyDeleteIntegration",
            "Effect": "Allow",
            "Action": [
                "redshift:ModifyIntegration",
                "redshift:DeleteIntegration"
```

```
            ],
            "Resource": "arn:aws:redshift:us-
east-1:111122223333:integration:<uuid>"
        },
        {
            "Sid": "AllowRedshiftCreateInboundIntegration",
            "Effect": "Allow",
            "Action": "redshift:CreateInboundIntegration",
            "Resource": "arn:aws:redshift:us-
east-1:111122223333:namespace:<uuid>"
        }
    ]
}
```

Encrypting DynamoDB integrations with a customer managed key

If you specify a custom KMS key rather than an AWS owned key when you create a DynamoDB zero-ETL integration, the key policy must provide the Amazon Redshift service principal access to the `CreateGrant` action. In addition, it must allow the requester account or role permission to run the `DescribeKey` and `CreateGrant` actions.

The following sample key policy statements demonstrate the permissions required in your policy. Some examples include context keys to further reduce the scope of permissions.

Sample key policy statements

The following policy statement allows the requester account or role to retrieve information about a KMS key.

```
{
   "Effect":"Allow",
   "Principal":{
      "AWS":"arn:aws:iam::{account-ID}:role/{role-name}"
   },
   "Action":"kms:DescribeKey",
   "Resource":"*"
}
```

The following policy statement allows the requester account or role to add a grant to a KMS key. The `kms:ViaService` condition key limits use of the KMS key to requests from Amazon Redshift.

```
{
   "Effect":"Allow",
   "Principal":{
      "AWS":"arn:aws:iam::{account-ID}:role/{role-name}"
   },
   "Action":"kms:CreateGrant",
   "Resource":"*",
   "Condition":{
      "StringEquals":{
         "kms:EncryptionContext:{context-key}":"{context-value}",
         "kms:ViaService":"redshift.{region}.amazonaws.com"
      },
      "ForAllValues:StringEquals":{
         "kms:GrantOperations":[
            "Decrypt",
            "GenerateDataKey",
            "CreateGrant"
         ]
      }
   }
}
```

The following policy statement allows the Amazon Redshift service principal to add a grant to a KMS key.

```
{
   "Effect":"Allow",
   "Principal":{
      "Service":"redshift.amazonaws.com"
   },
   "Action":"kms:CreateGrant",
   "Resource":"*",
   "Condition":{
      "StringEquals":{
         "kms:EncryptionContext:{context-key}":"{context-value}",
         "aws:SourceAccount":"{account-ID}"
      },
      "ForAllValues:StringEquals":{
         "kms:GrantOperations":[
            "Decrypt",
            "GenerateDataKey",
            "CreateGrant"
         ]
```

```
      },
      "ArnLike":{
         "aws:SourceArn":"arn:aws:*:{region}:{account-ID}:integration:*"
      }
   }
}
```

For more information, see Creating a key policy in the *AWS Key Management Service Developer Guide*.

Encryption context

When you encryption a zero-ETL integration, you can add key-value pairs as an **Additional encryption context**. You might want to add these key-value pairs to add additional contextual information about the data being replicated. For more information, see Encryption context in the *AWS Key Management Service Developer Guide*.

Amazon Redshift adds the following encryption context pairs in addition to any that you add:

- `aws:redshift:integration:arn` - `IntegrationArn`
- `aws:servicename:id` - `Redshift`

This reduces the overall number of pairs that you can add from 8 to 6, and contributes to the overall character limit of the grant constraint. For more information, see Using grant constraints in the *AWS Key Management Service Developer Guide*.

Create a zero-ETL integration with applications

In this step, you create a zero-ETL integration with applications with Amazon Redshift.

To create a zero-ETL integration with applications with Amazon Redshift

1. From the Amazon Redshift console: Create and configure a target Amazon Redshift data warehouse.
   - From the AWS CLI or Amazon Redshift console: Turn on case sensitivity for your data warehouse.
   - From the Amazon Redshift console: Configure authorization for your Amazon Redshift data warehouse.

2. From the AWS Glue console: Creating an integration as described in the *AWS Glue Developer Guide*.
3. After the destination database has been created and data starts replicating, you can query and create materialized data for your replicated data. For more information, see Querying replicated data in Amazon Redshift.

For detailed information to create zero-ETL integrations with applications, see Zero-ETL integrations in the *AWS Glue Developer Guide*.

Creating destination databases in Amazon Redshift

To replicate data from your source into Amazon Redshift, you must create a database from your integration in Amazon Redshift.

Connect to your target Redshift Serverless workgroup or provisioned cluster and create a database with a reference to your integration identifier. This identifier is the value returned for `integration_id` when you query the SVV_INTEGRATION view.

> **Important**
>
> Before creating a database from your integration, your zero-ETL integration must be created and in the `Active` state on the Amazon Redshift console.

Before you can start replicating data from your source into Amazon Redshift, create a database from the integration in Amazon Redshift. You can either create the database using the Amazon Redshift console or the query editor v2.

Amazon Redshift console

1. In the left navigation pane, choose **Zero-ETL integrations**.
2. From the integration list, choose an integration.
3. If you're using a provisioned cluster, you must first connect to the database. Choose **Connect to database**. You can connect using a recent connection, or by creating a new connection.
4. To create a database from the integration, choose **Create database from integration**.

5. Enter a **Destination database name**. The **Integration ID** and **Data warehouse name** are pre-populated.

   For Aurora PostgreSQL, RDS for PostgreSQL, RDS for Oracle, or Oracle Database@AWS sources, enter the **Source named database** that you specified when creating your zero-ETL integration. In these cases, you can map a maximum of 100 source databases to Amazon Redshift databases.

6. Choose **Create database**.

Amazon Redshift query editor v2

1. Navigate to the Amazon Redshift console and choose **Query editor v2**.
2. In the left panel, choose your Amazon Redshift Serverless workgroup or Amazon Redshift provisioned cluster, and then connect to it.
3. To get the integration ID, navigate to the integration list on the Amazon Redshift console.

   Alternatively, run the following command to get the `integration_id` value:

   ```
   SELECT integration_id FROM SVV_INTEGRATION;
   ```

4. Then, run the following command to create the database. By specifying the integration ID, you create a connection between the database and your source.

   Substitute `integration_id` with the value returned by the previous command.

   ```
   CREATE DATABASE destination_db_name FROM INTEGRATION 'integration_id';
   ```

   For Aurora PostgreSQL, RDS for PostgreSQL, RDS for Oracle, or Oracle Database@AWS sources, you must also include a reference to the named database within the source instance or cluster that you specified when you created the integration. For example:

   ```
   CREATE DATABASE "destination_db_name" FROM INTEGRATION 'integration_id'
    DATABASE "named_db";
   ```

For more information about creating a database for a zero-ETL integration target, see CREATE DATABASE in the *Amazon Redshift Database Developer Guide*. You can use ALTER DATBASE to change database parameters such as REFRESH INTERVAL. For more information about altering a

database for a zero-ETL integration target, see ALTER DATABASE in the *Amazon Redshift Database Developer Guide*.

> **Note**
>
> Only your integration source can update data in the database you create from your integration. To change the schema of a table, run DDL or DML commands against tables in the source. You can run DDL and DML commands against tables in the source, but you can only run DDL commands and read-only queries on the destination database.

For information about viewing the status of a destination database, see Viewing zero-ETL integrations.

After creating a destination database, you can add data to your source. To add data to your source, see one of the following topics:

- For Aurora sources, see Add data to the source DB cluster in the *Amazon Aurora User Guide*.
- For Amazon RDS sources, see Add data to the source DB instance in the *Amazon RDS User Guide*.
- For DynamoDB sources, see Getting started with DynamoDB in the *Amazon DynamoDB Developer Guide*.
- For zero-ETL integrations with applications sources, see Zero-ETL integrations in the *AWS Glue Developer Guide*.

Querying replicated data in Amazon Redshift

After you add data to your source, it's replicated in near real time to the Amazon Redshift data warehouse, and it's ready for querying. For information about integration metrics and table statistics, see Metrics for zero-ETL integrations.

> **Note**
>
> As a database is the same as a schema in MySQL, MySQL database level maps to Amazon Redshift schema level. Note this mapping difference when you query data replicated from Aurora MySQL or RDS for MySQL.

To query replicated data

1. Navigate to the Amazon Redshift console and choose **Query editor v2**.
2. Connect to your Amazon Redshift Serverless workgroup or Amazon Redshift provisioned cluster and choose your database from the dropdown list.
3. Use a SELECT statement to select all replicated data from the schema and table that you created in the source. For case sensitivity, use double quotes (" ") for schema, table, and column names. For example:

   ```
   SELECT * FROM "schema_name"."table_name";
   ```

   You can also query the data using the Amazon Redshift Data API.

Querying replicated data with materialized views

You can create materialized views in your local Amazon Redshift database to transform data replicated through zero-ETL integrations. Connect to your local database and use cross-database queries to access the destination databases. You can use either fully qualified object names with the three-part notation (destination-database-name.schema-name.table-name) or create an external schema referencing the destination database-schema pair and use the two-part notation (external-schema-name.table-name). For more information on cross-database queries, see Querying data across databases.

Use the following example to create and insert sample data into the *sales_zetl* and *event_zetl* tables from the source *tickit_zetl*. The tables are replicated into the Amazon Redshift database *zetl_int_db*.

```
CREATE TABLE sales_zetl (
        salesid integer NOT NULL primary key,
        eventid integer NOT NULL,
        pricepaid decimal(8, 2)
);

CREATE TABLE event_zetl (
        eventid integer NOT NULL PRIMARY KEY,
        eventname varchar(200)
);

INSERT INTO sales_zetl VALUES(1, 1, 3.33);
```

```
INSERT INTO sales_zetl VALUES(2, 2, 4.44);
INSERT INTO sales_zetl VALUES(3, 2, 5.55);

INSERT INTO event_zetl VALUES(1, "Event 1");
INSERT INTO event_zetl VALUES(2, "Event 2");
```

You can create a materialized view to get total sales per event using the three-part notation:

```
--three part notation zetl-database-name.schema-name.table-name
CREATE MATERIALIZED VIEW mv_transformed_sales_per_event_3p
AUTO REFRESH YES
AS
(SELECT eventname, sum(pricepaid) as total_price
FROM  zetl_int_db.tickit_zetl.sales_zetl S, zetl_int_db.tickit_zetl.event_zetl E
WHERE S.eventid = E.eventid
GROUP BY 1);
```

You can create a materialized view to get total sales per event using the two-part notation:

```
--two part notation external-schema-name.table-name notation
CREATE EXTERNAL schema ext_tickit_zetl
FROM REDSHIFT
DATABASE zetl_int_db
SCHEMA tickit_zetl;

CREATE MATERIALIZED VIEW mv_transformed_sales_per_event_2p
AUTO REFRESH YES
AS
(
    SELECT eventname, sum(pricepaid) as total_price
    FROM  ext_tickit_zetl.sales_zetl S, ext_tickit_zetl.event_zetl E
    WHERE S.eventid = E.eventid
    GROUP BY 1
);
```

To view the materialized views you created, use the following example.

```
SELECT * FROM mv_transformed_sales_per_event_3p;

+-----------+-------------+
| eventname | total_price |
+-----------+-------------+
```

```
| Event 1   | 3.33        |
| Event 2   | 9.99        |
+-----------+-------------+

SELECT * FROM mv_transformed_sales_per_event_2p;

+-----------+-------------+
| eventname | total_price |
+-----------+-------------+
| Event 1   | 3.33        |
| Event 2   | 9.99        |
+-----------+-------------+
```

Querying replicated data from DynamoDB

When you replicate data from Amazon DynamoDB to a Amazon Redshift database, it is stored in a materialized view in a column of SUPER data type.

For this example, the following data is stored in DynamoDB.

```
{
    "key1": {
        "S": "key_1"
    },
    "key2": {
        "N": 0
    },
    "payload": {
        "L": [
            {
                "S": "sale1"
            },
            {
                "S": "sale2"
            },
        ]
    },
}
```

The Amazon Redshift materialized view is defined as the following.

```
CREATE MATERIALIZED VIEW mv_sales
```

```
                    BACKUP NO
                    AUTO REFRESH YES
                    AS
                    SELECT "value"."payload"."L"[0]."S"::VARCHAR AS first_payload
                    FROM public.sales;
```

To view the data in the materialized view run an SQL command.

```
SELECT first_payload FROM mv_sales;
```

Viewing zero-ETL integrations

You can view your zero-ETL integrations from the Amazon Redshift console. Here you can view its configuration information and current status, and open screens to query and share data.

Amazon Redshift console

To view the details of a zero-ETL integration

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. From the left navigation pane, choose either the **Serverless** or **Provisioned clusters** dashboard. Then, choose **Zero-ETL integrations**.
3. Select the zero-ETL integration that you want to view. For each integration, the following information is provided:

   - **Integration ID** is the identifier returned when the integration is created.
   - **Status** can be one of the following:
     - `Active` – The zero-ETL integration is sending transactional data to the target Amazon Redshift data warehouse.
     - `Syncing` – The zero-ETL integration has encountered a recoverable error and is reseeding data. Affected tables aren't available for querying in Amazon Redshift until they finish resyncing.
     - `Failed` – The zero-ETL integration encountered an unrecoverable event or error that can't be fixed. You must delete and recreate the zero-ETL integration.
     - `Creating` – The zero-ETL integration is being created.
     - `Deleting` – The zero-ETL integration is being deleted.

  - `Needs attention` – The zero-ETL integration encountered an event or error that requires manual intervention to resolve it. To fix the issue, follow the steps in the error message.
- **Source type** is the type of source data replicating to the target. Types can specify other database managers, such as Aurora MySQL-Compatible Edition, Amazon Aurora PostgreSQL, RDS for MySQL, and from applications (`GlueSAAS`).
- **Source ARN** is the ARN of the source data. For most sources this is the ARN of the source database or table. For zero-ETL integration with applications sources, this is the ARN of the AWS Glue connection object.
- **Target** is the namespace of the Amazon Redshift data warehouse receiving source data.
- **Database** can be one of the following:
  - `No database` – There is no destination database for the integration.
  - `Creating` – Amazon Redshift is creating the destination database for the integration.
  - `Active` – Data is being replicated from the integration source to Amazon Redshift.
  - `Error` – There is an error with the integration.
  - `Recovering` – The integration is recovering after the data warehouse restarted.
  - `Resyncing` – Amazon Redshift is resynchronizing the tables in the integration.
- **Target type** is the type of Amazon Redshift data warehouse.
- **Creation date** is the date and time (UTC) when the integration was created.

> **Note**
>
> To view integration details for a data warehouse, choose the details page for your provisioned cluster or serverless namespace and then choose the **Zero-ETL integrations** tab.

From the **Zero-ETL integrations** list, you can choose **Query data** to jump to Amazon Redshift query editor v2. The Amazon Redshift target database has the enable_case_sensitive_identifier parameter enabled. When you write SQL, you might need to surround schemas, tables, and column names with double quotes ("<name>"). For more information about querying data in your Amazon Redshift data warehouse, see Querying a database using the query editor v2.

From the **Zero-ETL integrations** list, you can choose **Share data** to create a datashare. To create a datashare for the Amazon Redshift database, follow the instructions on the **Create datashare** page. Before you can share data in your Amazon Redshift database, you must first create a destination database. For more information about data sharing, see Data sharing concepts for Amazon Redshift.

To refresh your integration, you can use the ALTER DATABASE command. Doing so replicates all of the data from your integration source into your destination database. The following example refreshes all synced and failed tables within your zero-ETL integration.

```
ALTER DATABASE sample_integration_db INTEGRATION REFRESH ALL tables;
```

AWS CLI

To describe an Amazon DynamoDB zero-ETL integration with Amazon Redshift using the AWS CLI, use the `describe-integrations` command with the following options:

- `integration-arn` – Specify the ARN of the DynamoDB integration to describe.
- `integration-name` – Specify an optional filter that specifies one or more resources to return.

The follow example describes an integration by providing the integration ARN.

```
aws redshift describe-integrations

{
    "Integrations": [
        {
            "Status": "failed",
            "IntegrationArn": "arn:aws:redshift:us-
east-1:123456789012:integration:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Errors": [
                {
                    "ErrorCode": "INVALID_TABLE_PERMISSIONS",
                    "ErrorMessage": "Redshift does not have sufficient access on the
 table key. Refer to the Amazon DynamoDB Developer Guide."
                }
            ],
            "Tags": [],
            "CreateTime": "2023-11-09T00:32:46.444Z",
```

```
            "KMSKeyId": "arn:aws:kms:us-east-1:123456789012:key/a1b2c3d4-5678-90ab-
cdef-EXAMPLE33333",
            "TargetArn": "arn:aws:redshift:us-
east-1:123456789012:namespace:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "IntegrationName": "ddb-to-provisioned-02",
            "SourceArn": "arn:aws:dynamodb:us-east-1:123456789012:table/mytable"
        }
    ]
}
```

You can also filter the results of `describe-integrations` by the `integration-arn`, `source-arn`, `source-types`, or `status`. For more information, see describe-integrations in the *Amazon Redshift CLI Guide*.

History mode

With history mode, you can configure your zero-ETL integrations to track every version (including updates and deletes) of your records in source tables, directly in Amazon Redshift. You can run advanced analytics on all your data, such as, run a historical analysis, build look-back reports, perform trend analysis, and send incremental updates to downstream applications built on top of Amazon Redshift. History mode is supported with multiple Amazon Redshift zero-ETL integrations, including Amazon Aurora MySQL, Amazon Aurora PostgreSQL, Amazon RDS for MySQL, and Amazon DynamoDB. History mode is also supported by several applications, such as Salesforce, SAP, ServiceNow, and Zendesk.

You can turn on and off history mode for your zero-ETL integrations from the Amazon Redshift console (https://console.aws.amazon.com/redshiftv2/). Use history mode to keep track of records that have been deleted or modified in the source of integration. Tracking happens in the target Amazon Redshift data warehouse. Turning on history mode does not impact the performance of regular analytic queries on these tables.

After you turn on history mode, tables that you drop within the source won't be dropped in Amazon Redshift. Instead, tables will appear in a `DroppedSource` state and you can still query these tables. You can also still use the DROP and RENAME commands with regular SQL.

If you want to reuse the same table name on the source, you must DROP or RENAME the corresponding `DroppedState` table before it can be replicated to Amazon Redshift. Make sure to do so before you create the table on the source.

For information about what to consider when using history mode, see Considerations when using history mode on the target.

To manage history mode for a zero-ETL integration

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. From the left navigation pane, choose either the **Serverless** or **Provisioned clusters** dashboard. Then, choose **Zero-ETL integrations**.
3. Select the zero-ETL integration that you want to manage, choose **Manage history mode**. The **Manage history mode** window is displayed.
4. You can **Turn off** or **Turn on** history mode for a target table that is replicated from a source type that has a single source table, such as, Amazon DynamoDB. When the zero-ETL integration has multiple target tables possible, you can **Turn off for all existing and future tables**, **Turn on for all existing and future tables**, or **Manage history mode for individual tables**. The default is history mode `off` when the zero-ETL integration is created.

   When history mode is turned on, the following columns are added to your target table to keep track of changes in the source. History mode on increases monthly usage and cost because Amazon Redshift doesn't delete any records in the target tables. Any source record that is deleted or changed creates a new record in the target, resulting in more total rows in the target with multiple record versions. Records are not deleted from the target table when deleted or modified in the source. You can manage target tables by deleting inactive records.

Column name	Data type	Description
_record_is_active	Boolean	Indicates if a record in the target is currently active in the source. True indicates the record is active.
_record_create_time	Timestamp	Starting time (UTC) when the source record is active.
_record_delete_time	Timestamp	Ending time (UTC) when the source record is updated or deleted.

You can delete inactive records from a history mode table by filtering on records where the column `_record_is_active` is false. The following SQL DELETE command deletes inactive records from a table where the id column is less than or equal to 100. After you delete records, when automatic vacuum delete runs, storage for the deleted records is reclaimed.

```
DELETE FROM myschema.mytable where not _record_is_active AND id <= 100;
```

> **Note**
>
> This DELETE applies only to rows whose inactive state zero-ETL has already persisted in the table (usually within an hour, but can be longer for a table with infrequent changes). Rows that zero-ETL is still persisting in the background are left intact, so the DELETE can remove fewer rows than you expect. Run it again later to remove them. The `_record_is_active` filter is optional, because a DELETE on a table with history mode on removes only inactive rows.

When history mode is turned `off`, Amazon Redshift makes a copy of your table in the target database with active records and without the added history columns. Amazon Redshift renames your table to *`table-name`*_historical_*`timestamp`* for your use. You can drop this copy of your table if you no longer need it. You can rename these tables using the ALTER TABLE command. For example:

```
ALTER TABLE [schema-name.]table-name_historical_timestamp RENAME TO new_table_name;
```

For more information, see ALTER TABLE in the *Amazon Redshift Database Developer Guide*.

You can also manage history mode using SQL commands CREATE DATABASE and ALTER DATABASE. For more information about how to set HISTORY_MODE, see CREATE DATABASE and ALTER DATABASE in the *Amazon Redshift Database Developer Guide*.

Sharing your data in Amazon Redshift

After you add data to the source, it's immediately replicated into Amazon Redshift and ready to be shared by creating datashares.

To share data, you must create a destination database first.

To share data in Amazon Redshift Serverless using the Amazon Redshift console

1. In the Amazon Redshift console, in the left navigation pane, choose **Amazon Redshift Serverless > Serverless dashboard**.
2. From the left navigation pane, choose **Zero-ETL integrations**.
3. Choose **Share data**.
4. On the create datashare page, follow the steps in Creating datashares.

To share data in Amazon Redshift provisioned clusters using the Amazon Redshift console

1. In the Amazon Redshift console, in the left navigation pane, choose **Provisioned clusters dashboard**.
2. From the left navigation pane, choose **Zero-ETL integrations**.
3. From the integration list, choose an integration.
4. On the integration details page, choose **Connect to database**.
5. On the **Connection to database** page, you can either create a new connection or use a recent connection. Make sure that the connection is made to the destination database.
6. If you create a new connection, then enter a **Database name** for the database. Then, click **Connect**.
7. On the integration details page, choose **Share data**.
8. On the create datashare page, follow the steps in Creating datashares.

Monitoring zero-ETL integrations

You can monitor your zero-ETL integrations by querying the system views or with Amazon EventBridge.

Monitoring zero-ETL integrations with Amazon Redshift system views

You can monitor your zero-ETL integrations by querying the following system views in Amazon Redshift.

- SVV_INTEGRATION provides information about configuration details of zero-ETL integrations.

- SYS_INTEGRATION_ACTIVITY provides information about completed zero-ETL integrations.
- SVV_INTEGRATION_TABLE_MAPPING provides information about mapping metadata values from source to target.
- SVV_INTEGRATION_TABLE_STATE provides information about integration state.
- SYS_INTEGRATION_TABLE_ACTIVITY provides information about insert, delete, and update activity of integrations.
- SYS_INTEGRATION_TABLE_STATE_CHANGE provides information about table state change log for integrations.

Monitoring zero-ETL integrations with Amazon EventBridge

Amazon Redshift sends integration-related events to Amazon EventBridge. For a list of events and their corresponding event IDs, see Zero-ETL integration event notifications with Amazon EventBridge.

Metrics for zero-ETL integrations

You can use the metrics in the Amazon Redshift console and Amazon CloudWatch to learn about the health and performance of your zero-ETL integrations. You can adjust the metrics to display data for shorter or longer duration, or choose to view metrics in CloudWatch. To view the metrics for your integration on the Amazon Redshift console, choose **Zero-ETL integrations** in the left navigation pane and choose your integration ID.

Depending on the source data of zero-ETL integrations, Amazon Redshift provides metrics on the integration details page for an integration. Possible metrics include the following types:

- From the **Integration metrics** tab, graphs of the following are available:

Metric	Metric name in Amazon Redshift console	Description
`IntegrationLag`	`Lag`	The lag from the time data is committed to your source to the time when the data is available for queries in Amazon Redshift. Units: Seconds

Metric	Metric name in Amazon Redshift console	Description
		Dimensions: `IntegrationId` Update Frequency: 60 seconds

Metric	Metric name in Amazon Redshift console	Description
IntegrationState	State	The state of the integration, as one of the following numbers: • 0 – Waiting for the Amazon Redshift database to be created. • 1 – Creating the schema in the Amazon Redshift database. • 2 – Replicating. • 3 – Error. • 4 – Stopped. • 5 – Recovering after a pause or restart. • 8 – Waiting for the source side to be created. For integrations that replicate multiple databases, each database has its own state, and the metric reports the state that comes first in this order of priority: recovering (5), replicating (2), creating the schema (1), waiting for the Amazon Redshift database (0), waiting for the source side (8), error (3), and stopped (4). For example, if one database is replicating (2) and another is still waiting for its Amazon Redshift database to be created (0), the metric reports replicating (2). Units: Count Dimensions: IntegrationId Update Frequency: 60 seconds

Metric	Metric name in Amazon Redshift console	Description
IntegrationNumTablesReplicated	Tables replicated	The number of tables that have been replicated from your source database to Amazon Redshift. Units: Count Dimensions: IntegrationId Update Frequency: 60 seconds
IntegrationNumTablesFailedReplication	Tables failed	The number of tables that failed replication. Units: Count Dimensions: IntegrationId Update Frequency: 60 seconds
IntegrationDataTransferred	Data transferred	The amount of data transferred in logical bytes. Units: Bytes Dimensions: IntegrationId Update Frequency: 60 seconds
IntegrationLatestDetectedChange	Latest detected change	The time, in Unix epoch seconds, when the integration last staged a source change in the replication queue. Units: Seconds Dimensions: IntegrationId Update Frequency: 60 seconds

Metric	Metric name in Amazon Redshift console	Description
IntegrationLatestAppliedChange	Latest applied change	The time, in Unix epoch seconds, when the integration last completed ingestion on Amazon Redshift. Units: Seconds Dimensions: IntegrationId Update Frequency: 60 seconds
IntegrationDuplicateRowsDetected	Duplicate rows detected	The number of tables with duplicate rows identified during the reporting interval. Units: Count Dimensions: IntegrationId Update Frequency: 60 seconds
IntegrationAutoRemediationTriggered	Auto-remediation triggered	The number of tables that auto-remediation moved to resynchronization during the reporting interval. Units: Count Dimensions: IntegrationId Update Frequency: 60 seconds

Note

For integrations that replicate multiple databases, the IntegrationLatestDetectedChange and IntegrationLatestAppliedChange metrics report the minimum value across all databases. This represents the least recently updated database. Per-database values are available in the

`latest_detected_change_time` and `latest_applied_change_time` columns of SVV_INTEGRATION.

- From the **Table statistics** tab, you can view the list of tables that are currently active or have errors. The statistics on this tab are as follows (depending on source type):
  - **Schema name** – The name of the schema that the table is in.
  - **Table name** – The name of the table in the source database.
  - **Status** – The status of the table. Possible values include `Synced`, `Failed`, `Deleted`, `Resync Required`, and `Resync Initiated`.
  - **Database** – The Amazon Redshift database the table is in.
  - **Last updated** – The date and time (UTC) when the last update was made to the table.
  - **Table row count** – The number of rows in the table.
  - **Table size** – The size of the table.

You can also view a graph of the number of **Rows** inserted, deleted, and updated for the selected timeframe.

Modify a zero-ETL integration for DynamoDB

In this step, you modify an DynamoDB zero-ETL integration with Amazon Redshift.

Amazon Redshift console

To modify an Amazon DynamoDB zero-ETL integration with Amazon Redshift using the Amazon Redshift console

1. From the Amazon Redshift console, choose **Zero-ETL integrations**. On the pane with the list of zero-ETL integrations, then choose the DynamoDB integration that you want to modify.
2. Choose **Edit** and make modifications to the **Integration name** or **Description**.
3. Choose **Save changes** to save your changes.

AWS CLI

To modify an Amazon DynamoDB zero-ETL integration with Amazon Redshift using the AWS CLI, use the `modify-integration` command with the following options:

- `integration-arn` – Specify the ARN of the DynamoDB integration to modify.
- `integration-name` – Specify a new name for the integration.
- `description` – Specify a new description for the integration.

The follow example modifies an integration by providing the integration ARN, new description, and new name.

```
aws redshift modify-integration \
--integration-arn arn:aws:redshift:us-
east-1:123456789012:integration:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
--description "Test modify description and name together." \
--integration-name "updated-integration-name-2"

{
    "IntegrationArn": "arn:aws:redshift:us-
east-1:123456789012:integration:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "IntegrationName": "updated-integration-name-2",
    "SourceArn": "arn:aws:dynamodb:us-east-1:123456789012:table/ddb-temp-test-table-
table",
    "SourceType": "dynamodb",
    "TargetArn": "arn:aws:redshift:us-
east-1:123456789012:namespace:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "Status": "active",
    "Errors": [],
    "CreateTime": "2024-09-19T18:06:33.555Z",
    "Description": "Test modify description and name together.",
    "KMSKeyId": "arn:aws:kms:us-east-1:123456789012:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE33333",
    "AdditionalEncryptionContext": {},
    "Tags": []
}
```

Delete a zero-ETL integration for DynamoDB

When you delete an integration, the target data warehouse retains any previously replicated data. You can continue to share and query this data. However, new data in the source will not replicate to the target.

In this step, you delete an DynamoDB zero-ETL integration with Amazon Redshift.

Amazon Redshift console

To delete an Amazon DynamoDB zero-ETL integration with Amazon Redshift using the Amazon Redshift console

1. From the Amazon Redshift console, choose **Zero-ETL integrations**. On the pane with the list of zero-ETL integrations, then choose the DynamoDB integration that you want to delete.
2. Choose **Delete** and provide the requested information.
3. Choose **Delete** to delete the zero-ETL integration.

AWS CLI

To delete an Amazon DynamoDB zero-ETL integration with Amazon Redshift using the AWS CLI, use the `delete-integration` command with the following options:

- `integration-arn` – Specify the ARN of the DynamoDB integration to delete.

The follow example deletes an integration by providing the integration ARN.

```
aws redshift delete-integration \
--integration-arn arn:aws:redshift:us-
east-1:123456789012:integration:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111

{
    "IntegrationArn": "arn:aws:redshift:us-
east-1:123456789012:integration:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "IntegrationName": "updated-integration-name-2",
    "SourceArn": "arn:aws:dynamodb:us-east-1:123456789012:table/tidal-ddb-ddb-temp-
test-table-table",
    "SourceType": "dynamodb",
    "TargetArn": "arn:aws:redshift:us-
east-1:123456789012:namespace:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "Status": "deleting",
    "Errors": [],
    "CreateTime": "2024-09-19T18:06:33.555Z",
    "Description": "Test modify description and name together.",
    "KMSKeyId": "arn:aws:kms:us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "AdditionalEncryptionContext": {},
```

```
    "Tags": []
}
```

Supported Regions for zero-ETL integrations

Zero-ETL integration is a fully managed solution that makes transactional and operational data available in Amazon Redshift from multiple operational and transactional sources, as well as enterprise applications. This page lists the available Regions for each supported source.

Aurora MySQL

The following Regions and engine versions are available for Aurora MySQL zero-ETL integrations with Amazon Redshift.

Region	Aurora MySQL
Africa (Cape Town)	Available
Asia Pacific (Hong Kong)	Available
Asia Pacific (Tokyo)	Available
Asia Pacific (Seoul)	Available
Asia Pacific (Osaka)	Available
Asia Pacific (Mumbai)	Available
Asia Pacific (Hyderabad)	Available
Asia Pacific (Singapore)	Available
Asia Pacific (Sydney)	Available
Asia Pacific (Jakarta)	Available
Asia Pacific (Melbourne)	Available
Asia Pacific (Malaysia)	Not available

Region	Aurora MySQL
Canada (Central)	Available
Canada West (Calgary)	Available
China (Beijing)	Available
China (Ningxia)	Available
Europe (Frankfurt)	Available
Europe (Zurich)	Available
Europe (Stockholm)	Available
Europe (Milan)	Available
Europe (Spain)	Available
Europe (Ireland)	Available
Europe (London)	Available
Europe (Paris)	Available
Israel (Tel Aviv)	Available
Middle East (UAE)	Available
Middle East (Bahrain)	Available
South America (São Paulo)	Available
US East (N. Virginia)	Available
US East (Ohio)	Available
US West (N. California)	Available
US West (Oregon)	Available

Region	Aurora MySQL
AWS GovCloud (US-East)	Not available
AWS GovCloud (US-West)	Not available

Aurora PostgreSQL

The following Regions are available for Aurora PostgreSQL zero-ETL integrations with Amazon Redshift.

Region	Aurora PostgreSQL
Africa (Cape Town)	Available
Asia Pacific (Hong Kong)	Available
Asia Pacific (Tokyo)	Available
Asia Pacific (Seoul)	Available
Asia Pacific (Osaka)	Available
Asia Pacific (Mumbai)	Available
Asia Pacific (Hyderabad)	Available
Asia Pacific (Singapore)	Available
Asia Pacific (Sydney)	Available
Asia Pacific (Jakarta)	Available
Asia Pacific (Melbourne)	Available
Asia Pacific (Malaysia)	Available
Canada (Central)	Available
Canada West (Calgary)	Available

Region	Aurora PostgreSQL
China (Beijing)	Available
China (Ningxia)	Available
Europe (Frankfurt)	Available
Europe (Zurich)	Available
Europe (Stockholm)	Available
Europe (Milan)	Available
Europe (Spain)	Available
Europe (Ireland)	Available
Europe (London)	Available
Europe (Paris)	Available
Israel (Tel Aviv)	Available
Middle East (UAE)	Available
Middle East (Bahrain)	Available
South America (São Paulo)	Available
US East (N. Virginia)	Available
US East (Ohio)	Available
US West (N. California)	Available
US West (Oregon)	Available
AWS GovCloud (US-East)	Not available
AWS GovCloud (US-West)	Not available

Amazon DynamoDB

The following Regions are available for DynamoDB zero-ETL integrations with Amazon Redshift.

Region	DynamoDB
Africa (Cape Town)	Available
Asia Pacific (Hong Kong)	Available
Asia Pacific (Taipei)	Available
Asia Pacific (Tokyo)	Available
Asia Pacific (Seoul)	Available
Asia Pacific (Osaka)	Available
Asia Pacific (Mumbai)	Available
Asia Pacific (Hyderabad)	Available
Asia Pacific (Singapore)	Available
Asia Pacific (Sydney)	Available
Asia Pacific (Jakarta)	Available
Asia Pacific (Melbourne)	Available
Asia Pacific (Malaysia)	Available
Asia Pacific (Thailand)	Available
Canada (Central)	Available
Canada West (Calgary)	Available
China (Beijing)	Available
China (Ningxia)	Available

Region	DynamoDB
Europe (Frankfurt)	Available
Europe (Zurich)	Available
Europe (Stockholm)	Available
Europe (Milan)	Available
Europe (Spain)	Available
Europe (Ireland)	Available
Europe (London)	Available
Europe (Paris)	Available
Israel (Tel Aviv)	Available
Middle East (UAE)	Available
Middle East (Bahrain)	Available
Mexico (Central)	Available
South America (São Paulo)	Available
US East (N. Virginia)	Available
US East (Ohio)	Available
US West (N. California)	Available
US West (Oregon)	Available
AWS GovCloud (US-East)	Available
AWS GovCloud (US-West)	Available

Amazon RDS for MySQL

The following Regions are available for Amazon RDS for MySQL zero-ETL integrations with Amazon Redshift.

Region	RDS for MySQL
Africa (Cape Town)	Available
Asia Pacific (Hong Kong)	Available
Asia Pacific (Tokyo)	Available
Asia Pacific (Seoul)	Available
Asia Pacific (Osaka)	Available
Asia Pacific (Mumbai)	Available
Asia Pacific (Hyderabad)	Not available
Asia Pacific (Singapore)	Available
Asia Pacific (Sydney)	Available
Asia Pacific (Jakarta)	Not available
Asia Pacific (Melbourne)	Not available
Asia Pacific (Malaysia)	Not available
Canada (Central)	Available
Canada West (Calgary)	Not available
China (Beijing)	Not available
China (Ningxia)	Not available
Europe (Frankfurt)	Available
Europe (Zurich)	Not available

Region	RDS for MySQL
Europe (Stockholm)	Available
Europe (Milan)	Available
Europe (Spain)	Not available
Europe (Ireland)	Available
Europe (London)	Available
Europe (Paris)	Available
Israel (Tel Aviv)	Not available
Middle East (UAE)	Not available
Middle East (Bahrain)	Available
South America (São Paulo)	Available
US East (N. Virginia)	Available
US East (Ohio)	Available
US West (N. California)	Available
US West (Oregon)	Available
AWS GovCloud (US-East)	Not available
AWS GovCloud (US-West)	Not available

Enterprise applications

The following Regions are available for enterprise application zero-ETL integrations with Amazon Redshift.

Region	Enterprise applications
Africa (Cape Town)	Not available
Asia Pacific (Hong Kong)	Available
Asia Pacific (Tokyo)	Available
Asia Pacific (Seoul)	Available
Asia Pacific (Osaka)	Not available
Asia Pacific (Mumbai)	Not available
Asia Pacific (Hyderabad)	Not available
Asia Pacific (Singapore)	Available
Asia Pacific (Sydney)	Available
Asia Pacific (Jakarta)	Not available
Asia Pacific (Melbourne)	Not available
Asia Pacific (Malaysia)	Not available
Canada (Central)	Available
Canada West (Calgary)	Not available
China (Beijing)	Not available
China (Ningxia)	Not available
Europe (Frankfurt)	Available
Europe (Zurich)	Not available
Europe (Stockholm)	Available
Europe (Milan)	Not available

Region	Enterprise applications
Europe (Spain)	Not available
Europe (Ireland)	Available
Europe (London)	Available
Europe (Paris)	Not available
Israel (Tel Aviv)	Not available
Middle East (UAE)	Not available
Middle East (Bahrain)	Not available
South America (São Paulo)	Available
US East (N. Virginia)	Available
US East (Ohio)	Available
US West (N. California)	Not available
US West (Oregon)	Available
AWS GovCloud (US-East)	Not available
AWS GovCloud (US-West)	Not available

Troubleshooting zero-ETL integrations

Use the following sections to help troubleshoot problems that you have with zero-ETL integrations.

Troubleshooting zero-ETL integrations with Aurora MySQL

Use the following information to troubleshoot common issues with zero-ETL integrations with Aurora MySQL.

Topics

- Creation of the integration failed

- Tables don't have primary keys
- Aurora MySQL tables aren't replicating to Amazon Redshift
- Unsupported data types in tables
- Data manipulation language commands failed
- Tracked changes between data sources don't match
- Authorization failed
- Number of tables is more than 100K or the number of schemas is more than 4950
- Amazon Redshift can't load data
- Workgroup parameter settings are incorrect
- Database isn't created to activate a zero-ETL integration
- Table is in the Resync Required or Resync Initiated state
- Integration lag growing

Creation of the integration failed

If the creation of the zero-ETL integration failed, the status of the integration is `Inactive`. Make sure that the following are correct for your source Aurora DB cluster:

- You created your cluster in the Amazon RDS console.
- Your source Aurora DB cluster is running a supported version. For a list of supported versions, see Supported Regions and Aurora DB engines for zero-ETL integrations with Amazon Redshift. To validate this, go to the **Configuration** tab for the cluster and check the **Engine version**.
- You correctly configured binlog parameter settings for your cluster. If your Aurora MySQL binlog parameters are set incorrectly or not associated with the source Aurora DB cluster, creation fails. See Configure DB cluster parameters.

In addition, make sure the following are correct for your Amazon Redshift data warehouse:

- Case sensitivity is turned on. See Turn on case sensitivity for your data warehouse.
- You added the correct authorized principal and integration source for your namespace. See Configure authorization for your Amazon Redshift data warehouse.

Tables don't have primary keys

In the destination database, one or more of the tables don't have a primary key and can't be synchronized.

To resolve this issue, go to the **Table statistics** tab on the integration details page or use SVV_INTEGRATION_TABLE_STATE to view the failed tables. You can add primary keys to the tables and Amazon Redshift will resynchronize the tables. Alternatively, although not recommended, you can drop these tables on Aurora and create tables with a primary key. For more information, see Amazon Redshift best practices for designing tables.

Aurora MySQL tables aren't replicating to Amazon Redshift

If you don't see one or more tables reflected in Amazon Redshift, you can run the following command to resynchronize them. Replace *dbname* with the name of your Amazon Redshift database. And, replace *table1* and *table2* with the names of the tables to be synchronized.

```
ALTER DATABASE dbname INTEGRATION REFRESH TABLES table1, table2;
```

For more information, see see ALTER DATABASE in the *Amazon Redshift Database Developer Guide*.

Your data might not be replicating because one or more of your source tables doesn't have a primary key. The monitoring dashboard in Amazon Redshift displays the status of these tables as `Failed`, and the status of the overall zero-ETL integration changes to `Needs attention`. To resolve this issue, you can identify an existing key in your table that can become a primary key, or you can add a synthetic primary key. For detailed solutions, see Handle tables without primary keys while creating Amazon Aurora MySQL or RDS for MySQL zero-ETL integrations with Amazon Redshift. in the *AWS Database Blog*.

Also confirm that if your target is an Amazon Redshift cluster, that the cluster is not paused.

Unsupported data types in tables

In the database that you created from the integration in Amazon Redshift and in which data is replicated from the Aurora DB cluster, one or more of the tables have unsupported data types and can't be synchronized.

To resolve this issue, go to the **Table statistics** tab on the integration details page or use SVV_INTEGRATION_TABLE_STATE to view the failed tables. Then, remove these tables and recreate

new tables on Amazon RDS. For more information on unsupported data types, see Data type differences between Aurora and Amazon Redshift databases in the *Amazon Aurora User Guide*.

Data manipulation language commands failed

Amazon Redshift could not run DML commands on the Redshift tables. To resolve this issue, use SVV_INTEGRATION_TABLE_STATE to view the failed tables. Amazon Redshift automatically resynchronizes the tables to resolve this error.

Tracked changes between data sources don't match

This error occurs when changes between Amazon Aurora and Amazon Redshift don't match, leading to the integration entering a `Failed` state.

To resolve this, delete the zero-ETL integration and create it again in Amazon RDS. For more information, see Creating zero-ETL integrations and Deleting zero-ETL integrations.

Authorization failed

Authorization failed because the source Aurora DB cluster was removed as an authorized integration source for the Amazon Redshift data warehouse.

To resolve this issue, delete the zero-ETL integration and create it again on Amazon RDS. For more information, see Creating zero-ETL integrations and Deleting zero-ETL integrations.

Number of tables is more than 100K or the number of schemas is more than 4950

For a destination data warehouse, the number of tables is more than 100K or the number of schemas is more than 4950. Amazon Aurora can't send data to Amazon Redshift. The number of tables and schemas exceeds the set limit. To resolve this issue, remove any unnecessary schemas or tables from the source database.

Amazon Redshift can't load data

Amazon Redshift can't load data to the zero-ETL integration.

To resolve this issue, delete the zero-ETL integration on Amazon RDS and create it again. For more information, see Creating zero-ETL integrations and Deleting zero-ETL integrations.

Workgroup parameter settings are incorrect

Your workgroup doesn't have case sensitivity turned on.

To resolve this issue, go to the **Properties** tab on the integration details page, choose the parameter group, and turn on the case-sensitive identifier from the **Properties** tab. If you don't have an existing parameter group, create one with the case-sensitive identifier turned on. Then, create a new zero-ETL integration on Amazon RDS. For more information, see Creating zero-ETL integrations.

Database isn't created to activate a zero-ETL integration

There isn't a database created for the zero-ETL integration to activate it.

To resolve this issue, create a database for the integration. For more information, see Creating destination databases in Amazon Redshift.

Table is in the Resync Required or Resync Initiated state

Your table is in the **Resync Required** or **Resync Initiated** state.

To gather more detailed error information about why your table is in that state, use the SYS_LOAD_ERROR_DETAIL system view.

Integration lag growing

The integration lag of your zero-ETL integrations can grow if there is a heavy use of SAVEPOINT in your source database.

Troubleshooting zero-ETL integrations with Aurora PostgreSQL

Use the following information to troubleshoot common issues with zero-ETL integrations with Aurora PostgreSQL.

Topics

- Creation of the integration failed
- Tables don't have primary keys
- Aurora PostgreSQL tables aren't replicating to Amazon Redshift
- Unsupported data types in tables
- Data manipulation language commands failed
- Tracked changes between data sources don't match
- Authorization failed

- Number of tables is more than 100K or the number of schemas is more than 4950
- Amazon Redshift can't load data
- Workgroup parameter settings are incorrect
- Database isn't created to activate a zero-ETL integration
- Table is in the Resync Required or Resync Initiated state

Creation of the integration failed

If the creation of the zero-ETL integration failed, the status of the integration is `Inactive`. Make sure that the following are correct for your source Aurora DB cluster:

- You created your cluster in the Amazon RDS console.
- Your source Aurora DB cluster is running supported version. For a list of supported versions, see Supported Regions and Aurora DB engines for zero-ETL integrations with Amazon Redshift. To validate this, go to the **Configuration** tab for the cluster and check the **Engine version**.
- You correctly configured binlog parameter settings for your cluster. If your Aurora PostgreSQL binlog parameters are set incorrectly or not associated with the source Aurora DB cluster, creation fails. See Configure DB cluster parameters.

In addition, make sure the following are correct for your Amazon Redshift data warehouse:

- Case sensitivity is turned on. See Turn on case sensitivity for your data warehouse.
- You added the correct authorized principal and integration source for your endterm="zero-etl-using.redshift-iam.title"/>.

Tables don't have primary keys

In the destination database, one or more of the tables don't have a primary key and can't be synchronized.

To resolve this issue, go to the **Table statistics** tab on the integration details page or use SVV_INTEGRATION_TABLE_STATE to view the failed tables. You can add primary keys to the tables and Amazon Redshift will resynchronize the tables. Alternatively, although not recommended, you can drop these tables on Aurora and create tables with a primary key. For more information, see Amazon Redshift best practices for designing tables.

Aurora PostgreSQL tables aren't replicating to Amazon Redshift

If you don't see one or more tables reflected in Amazon Redshift, you can run the following command to resynchronize them. Replace *dbname* with the name of your Amazon Redshift database. And, replace *table1* and *table2* with the names of the tables to be synchronized.

```
ALTER DATABASE dbname INTEGRATION REFRESH TABLES table1, table2;
```

For more information, see see ALTER DATABASE in the *Amazon Redshift Database Developer Guide*.

Your data might not be replicating because one or more of your source tables doesn't have a primary key. The monitoring dashboard in Amazon Redshift displays the status of these tables as `Failed`, and the status of the overall zero-ETL integration changes to `Needs attention`. To resolve this issue, you can identify an existing key in your table that can become a primary key, or you can add a synthetic primary key. For detailed solutions, see Handle tables without primary keys while creating Amazon Aurora PostgreSQL zero-ETL integrations with Amazon Redshift. in the *AWS Database Blog*.

Also confirm that if your target is an Amazon Redshift cluster, that the cluster is not paused.

Unsupported data types in tables

In the database that you created from the integration in Amazon Redshift and in which data is replicated from the Aurora DB cluster, one or more of the tables have unsupported data types and can't be synchronized.

To resolve this issue, go to the **Table statistics** tab on the integration details page or use SVV_INTEGRATION_TABLE_STATE to view the failed tables. Then, remove these tables and recreate new tables on Amazon RDS. For more information on unsupported data types, see Data type differences between Aurora and Amazon Redshift databases in the *Amazon Aurora User Guide*.

Data manipulation language commands failed

Amazon Redshift could not run DML commands on the Redshift tables. To resolve this issue, use SVV_INTEGRATION_TABLE_STATE to view the failed tables. Amazon Redshift automatically resynchronizes the tables to resolve this error.

Tracked changes between data sources don't match

This error occurs when changes between Amazon Aurora and Amazon Redshift don't match, leading to the integration entering a `Failed` state.

To resolve this, delete the zero-ETL integration and create it again in Amazon RDS. For more information, see Creating zero-ETL integrations and Deleting zero-ETL integrations.

Authorization failed

Authorization failed because the source Aurora DB cluster was removed as an authorized integration source for the Amazon Redshift data warehouse.

To resolve this issue, delete the zero-ETL integration and create it again on Amazon RDS. For more information, see Creating zero-ETL integrations and Deleting zero-ETL integrations.

Number of tables is more than 100K or the number of schemas is more than 4950

For a destination data warehouse, the number of tables is more than 100K or the number of schemas is more than 4950. Amazon Aurora can't send data to Amazon Redshift. The number of tables and schemas exceeds the set limit. To resolve this issue, remove any unnecessary schemas or tables from the source database.

Amazon Redshift can't load data

Amazon Redshift can't load data to the zero-ETL integration.

To resolve this issue, delete the zero-ETL integration on Amazon RDS and create it again. For more information, see Creating zero-ETL integrations and Deleting zero-ETL integrations.

Workgroup parameter settings are incorrect

Your workgroup doesn't have case sensitivity turned on.

To resolve this issue, go to the **Properties** tab on the integration details page, choose the parameter group, and turn on the case-sensitive identifier from the **Properties** tab. If you don't have an existing parameter group, create one with the case-sensitive identifier turned on. Then, create a new zero-ETL integration on Amazon RDS. For more information, see Creating zero-ETL integrations.

Database isn't created to activate a zero-ETL integration

There isn't a database created for the zero-ETL integration to activate it.

To resolve this issue, create a database for the integration. For more information, see Creating destination databases in Amazon Redshift.

Table is in the Resync Required or Resync Initiated state

Your table is in the **Resync Required** or **Resync Initiated** state.

To gather more detailed error information about why your table is in that state, use the SYS_LOAD_ERROR_DETAIL system view.

Troubleshooting zero-ETL integrations with RDS for MySQL

Use the following information to troubleshoot common issues with zero-ETL integrations with RDS for MySQL.

Topics

- Creation of the integration failed
- Tables don't have primary keys
- RDS for MySQL tables aren't replicating to Amazon Redshift
- Unsupported data types in tables
- Data manipulation language commands failed
- Tracked changes between data sources don't match
- Authorization failed
- Number of tables is more than 100K or the number of schemas is more than 4950
- Amazon Redshift can't load data
- Workgroup parameter settings are incorrect
- Database isn't created to activate a zero-ETL integration
- Table is in the Resync Required or Resync Initiated state

Creation of the integration failed

If the creation of the zero-ETL integration failed, the status of the integration is `Inactive`. Make sure that the following are correct for your source RDS DB instance:

- You created your instance in the Amazon RDS console.
- Your source RDS DB instance is running a supported version of RDS for MySQL. For a list of supported versions, see Supported Regions and DB engines for Amazon RDS zero-ETL integrations with Amazon Redshift. To validate this, go to the **Configuration** tab for the instance and check the **Engine version**.

- You correctly configured binlog parameter settings for your instance. If your RDS for MySQL binlog parameters are set incorrectly or not associated with the source RDS DB instance, creation fails. See Configure DB instance parameters.

In addition, make sure the following are correct for your Amazon Redshift data warehouse:

- Case sensitivity is turned on. See Turn on case sensitivity for your data warehouse.
- You added the correct authorized principal and integration source for your namespace. See Configure authorization for your Amazon Redshift data warehouse.

Tables don't have primary keys

In the destination database, one or more of the tables don't have a primary key and can't be synchronized.

To resolve this issue, go to the **Table statistics** tab on the integration details page or use SVV_INTEGRATION_TABLE_STATE to view the failed tables. You can add primary keys to the tables and Amazon Redshift will resynchronize the tables. Alternatively, although not recommended, you can drop these tables on RDS and create tables with a primary key. For more information, see Amazon Redshift best practices for designing tables.

RDS for MySQL tables aren't replicating to Amazon Redshift

If you don't see one or more tables reflected in Amazon Redshift, you can run the following command to resynchronize them. Replace *dbname* with the name of your Amazon Redshift database. And, replace *table1* and *table2* with the names of the tables to be synchronized.

```
ALTER DATABASE dbname INTEGRATION REFRESH TABLES table1, table2;
```

For more information, see see ALTER DATABASE in the *Amazon Redshift Database Developer Guide*.

Your data might not be replicating because one or more of your source tables doesn't have a primary key. The monitoring dashboard in Amazon Redshift displays the status of these tables as `Failed`, and the status of the overall zero-ETL integration changes to `Needs attention`. To resolve this issue, you can identify an existing key in your table that can become a primary key, or you can add a synthetic primary key. For detailed solutions, see Handle tables without primary keys while creating Aurora MySQL-Compatible Edition or RDS for MySQL zero-ETL integrations with Amazon Redshift. in the *AWS Database Blog*.

Also confirm that if your target is an Amazon Redshift cluster, that the cluster is not paused.

Unsupported data types in tables

In the database that you created from the integration in Amazon Redshift and in which data is replicated from the RDS DB instance, one or more of the tables have unsupported data types and can't be synchronized.

To resolve this issue, go to the **Table statistics** tab on the integration details page or use SVV_INTEGRATION_TABLE_STATE to view the failed tables. Then, remove these tables and recreate new tables on Amazon RDS. For more information on unsupported data types, see Data type differences between RDS and Amazon Redshift databases in the *Amazon RDS User Guide*.

Data manipulation language commands failed

Amazon Redshift could not run DML commands on the Redshift tables. To resolve this issue, use SVV_INTEGRATION_TABLE_STATE to view the failed tables. Amazon Redshift automatically resynchronizes the tables to resolve this error.

Tracked changes between data sources don't match

This error occurs when changes between Amazon Aurora and Amazon Redshift don't match, leading to the integration entering a `Failed` state.

To resolve this, delete the zero-ETL integration and create it again in Amazon RDS. For more information, see Creating zero-ETL integrations and Deleting zero-ETL integrations.

Authorization failed

Authorization failed because the source RDS DB instance was removed as an authorized integration source for the Amazon Redshift data warehouse.

To resolve this issue, delete the zero-ETL integration and create it again on Amazon RDS. For more information, see Creating zero-ETL integrations and Deleting zero-ETL integrations.

Number of tables is more than 100K or the number of schemas is more than 4950

For a destination data warehouse, the number of tables is more than 100K or the number of schemas is more than 4950. Amazon Aurora can't send data to Amazon Redshift. The number of tables and schemas exceeds the set limit. To resolve this issue, remove any unnecessary schemas or tables from the source database.

Amazon Redshift can't load data

Amazon Redshift can't load data to the zero-ETL integration.

To resolve this issue, delete the zero-ETL integration on Amazon RDS and create it again. For more information, see Creating zero-ETL integrations and Deleting zero-ETL integrations.

Workgroup parameter settings are incorrect

Your workgroup doesn't have case sensitivity turned on.

To resolve this issue, go to the **Properties** tab on the integration details page, choose the parameter group, and turn on the case-sensitive identifier from the **Properties** tab. If you don't have an existing parameter group, create one with the case-sensitive identifier turned on. Then, create a new zero-ETL integration on Amazon RDS. For more information, see Creating zero-ETL integrations.

Database isn't created to activate a zero-ETL integration

There isn't a database created for the zero-ETL integration to activate it.

To resolve this issue, create a database for the integration. For more information, see Creating destination databases in Amazon Redshift.

Table is in the Resync Required or Resync Initiated state

Your table is in the **Resync Required** or **Resync Initiated** state.

To gather more detailed error information about why your table is in that state, use the SYS_LOAD_ERROR_DETAIL system view.

Troubleshooting zero-ETL integrations with DynamoDB

Use the following information to troubleshoot common issues with zero-ETL integrations with Amazon DynamoDB.

Topics

- Creation of the integration failed
- Unsupported data types in tables
- Unsupported table and attribute names
- Authorization failed

- Amazon Redshift can't load data
- Workgroup or cluster parameter settings are incorrect
- Database isn't created to activate a zero-ETL integration
- Point-in-time recovery (PITR) is not enabled on source DynamoDB table
- KMS key access denied
- Amazon Redshift does not have access to DynamoDB table key

Creation of the integration failed

If the creation of the zero-ETL integration failed, the status of the integration is `Inactive`. Make sure that the following are correct for your Amazon Redshift data warehouse and source DynamoDB table:

- Case sensitivity is turned on for your data warehouse. See Turn on case sensitivity in the *Amazon Redshift Management Guide*.
- You added the correct authorized principal and integration source for your namespace in Amazon Redshift. See Configure authorization for your Amazon Redshift data warehouse in the *Amazon Redshift Management Guide*.
- You added the correct resource-based policy to the source DynamoDB table. See Policies and permissions in IAM in the *IAM User Guide*.

Unsupported data types in tables

DynamoDB numbers are translated to DECIMAL(38,10) in Amazon Redshift. Numbers exceeding this precision range are automatically transformed to (38,10). Delete the integration and unify the number precisions, and then re-create the integration.

Unsupported table and attribute names

Amazon Redshift supports up to 127 character table and attribute names. If a long name, such as the DynamoDB table name or the partition key or sort key column name fails your integration, fix it by using a shorter name and re-create the integration.

Authorization failed

Authorization can fail when the source DynamoDB table is removed as an authorized integration source for the Amazon Redshift data warehouse.

To resolve this issue, delete the zero-ETL integration, and re-create it using Amazon DynamoDB.

Amazon Redshift can't load data

Amazon Redshift can't load data from a zero-ETL integration.

To resolve this issue, refresh the integration with ALTER DATABASE.

```
ALTER DATABASE sample_integration_db INTEGRATION REFRESH ALL TABLES
```

Workgroup or cluster parameter settings are incorrect

Your workgroup or cluster doesn't have case sensitivity turned on.

To resolve this issue, go to the **Properties** tab on the integration details page, choose the parameter group, and turn on the case-sensitive identifier from the **Properties** tab. If you don't have an existing parameter group, create one with the case-sensitive identifier turned on. Then, create a new zero-ETL integration on DynamoDB. See Turn on case sensitivity in the *Amazon Redshift Management Guide*.

Database isn't created to activate a zero-ETL integration

There isn't a database created for the zero-ETL integration to activate it.

To resolve this issue, create a database for the integration. See Creating destination databases in Amazon Redshift in the *Amazon Redshift Management Guide*.

Point-in-time recovery (PITR) is not enabled on source DynamoDB table

Enabling PITR is required for DynamoDB to export data. Ensure PITR is always enabled. If you ever turn off PITR while the integration is active, you'll need to follow instructions in the error message and refresh the integration using ALTER DATABASE.

```
ALTER DATABASE sample_integration_db INTEGRATION REFRESH ALL TABLES
```

KMS key access denied

The KMS key used for the source table or integration must be configured with sufficient permissions. For information about table encryption and decryption, see DynamoDB encryption at rest in the *Amazon DynamoDB Developer Guide*.

Amazon Redshift does not have access to DynamoDB table key

If the source table encryption is an AWS managed key, then switch to an AWS owned key or customer managed key. If the table is already encrypted with a customer managed key, ensure that the policy doesn't have any condition keys.

Troubleshooting zero-ETL integrations with applications

Use the following information to troubleshoot common issues with zero-ETL integrations with applications, such as, Salesforce, SAP, ServiceNow, and Zendesk.

Topics

- Creation of the integration failed
- Tables aren't replicating to Amazon Redshift
- Unsupported data types in tables
- Workgroup parameter settings are incorrect
- Database isn't created to activate a zero-ETL integration
- Table is in the Resync Required or Resync Initiated state

Creation of the integration failed

If the creation of the zero-ETL integration failed, the status of the integration is `Inactive`. Make sure that the following are correct for your Amazon Redshift data warehouse:

- Case sensitivity is turned on. See Turn on case sensitivity for your data warehouse.
- You added the correct authorized principal and integration source for your namespace. See Configure authorization for your Amazon Redshift data warehouse.

Tables aren't replicating to Amazon Redshift

In the destination database, one or more of the tables don't have a primary key and can't be synchronized.

To resolve this issue, go to the **Table statistics** tab on the integration details page or use SVV_INTEGRATION_TABLE_STATE to view the failed tables. You can add primary keys to the tables and Amazon Redshift will resynchronize the tables. You can run the following command to resynchronize them. Replace *`dbname`* with the name of your Amazon Redshift database. And, replace *`table1`* and *`table2`* with the names of the tables to be synchronized.

```
ALTER DATABASE dbname INTEGRATION REFRESH TABLES table1, table2;
```

For more information, see ALTER DATABASE in the *Amazon Redshift Database Developer Guide*.

Unsupported data types in tables

In the database that you created from the integration in Amazon Redshift and in which data is replicated from zero-ETL integrations with applications, one or more of the tables have unsupported data types and can't be synchronized.

To resolve this issue, go to the **Table statistics** tab on the integration details page or use SVV_INTEGRATION_TABLE_STATE to view the failed tables. Then, remove these tables and recreate new tables at the source. For more information, see see Zero-ETL integrations in the *AWS Glue Developer Guide*.

Workgroup parameter settings are incorrect

Your workgroup doesn't have case sensitivity turned on.

To resolve this issue, go to the **Properties** tab on the integration details page, choose the parameter group, and turn on the case-sensitive identifier from the **Properties** tab. If you don't have an existing parameter group, create one with the case-sensitive identifier turned on. Then, create a new zero-ETL integration. For more information, see see Zero-ETL integrations in the *AWS Glue Developer Guide*.

Database isn't created to activate a zero-ETL integration

There isn't a database created for the zero-ETL integration to activate it.

To resolve this issue, create a database for the integration. For more information, see Creating destination databases in Amazon Redshift.

Table is in the Resync Required or Resync Initiated state

Your table is in the **Resync Required** or **Resync Initiated** state.

To gather more detailed error information about why your table is in that state, use the SYS_LOAD_ERROR_DETAIL system view.

Query a database

To query databases hosted by your Amazon Redshift cluster, you have two options:

- Connect to your cluster and run queries on the AWS Management Console with the query editor.

  If you use the query editor on the Amazon Redshift console, you don't have to download and set up a SQL client application.
- Connect to your cluster through a SQL client tool, such as SQL Workbench/J.

  Amazon Redshift supports SQL client tools connecting through Java Database Connectivity (JDBC) and Open Database Connectivity (ODBC). Amazon Redshift doesn't provide or install any SQL client tools or libraries, so you must install them on your client computer or Amazon EC2 instance to use them. You can use most SQL client tools that support JDBC or ODBC drivers.

Note

When you write stored procedures, we recommend a best practice for securing sensitive values:
Don't hard code any sensitive information in stored procedure logic. For example, don't assign a user password in a CREATE USER statement in the body of a stored procedure. This poses a security risk, because hard-coded values can be recorded as schema metadata in catalog tables. Instead, pass sensitive values, such as passwords, as arguments to the stored procedure, by means of parameters.
For more information about stored procedures, see CREATE PROCEDURE and Creating stored procedures in Amazon Redshift. For more information about catalog tables, see System catalog tables.

Connecting to Amazon Redshift

You can connect to your database using the following syntax.

```
cluster-name.account-number.aws-region.redshift.amazonaws.com/database-name
```

The syntax elements are defined as follows.

- `cluster-name`

  Your cluster's name.

- `account-number`

  The unique identifier associated with your AWS account number in a given AWS Region. All clusters created by a given account in a given AWS Region have the same `account-number`.

- `aws-region`

  The code for the AWS Region that the cluster is in.

- `database-name`

  Your database's name.

For example, the following connection string specifies the `my-db` database in the `my-cluster` cluster in the us-east-1 AWS Region.

```
my-cluster.123456789012.us-east-1.redshift.amazonaws.com/my-db
```

Querying a database using the query editor v2

The query editor v2 is a separate web-based SQL client application that you use to author and run queries on your Amazon Redshift data warehouse. The query editor v2 is primarily used to edit and run queries, visualize results, and share your work with your team. With query editor v2, you can create databases, schemas, tables, and user-defined functions (UDFs). In a tree-view panel, for each of your databases, you can view its schemas. For each schema, you can view its tables, views, UDFs, and stored procedures. The query editor v2 is a replacement for the previous query editor.

> **Note**
>
> The query editor v2 is available in commercial AWS Regions. For a list of AWS Regions where the query editor v2 is available, see the endpoints listed for Redshift query editor v2 in the *Amazon Web Services General Reference*.

For a demo of query editor v2, watch the following video. Amazon Redshift query editor v2.

For a demo of data analysis, watch the following video. Data analysis using Amazon Redshift query editor v2.

For a demo of using query editor v2 running multiple queries with either an isolated or shared connection, watch the following video. Concurrent Query Execution using Query Editor v2.

The query editor v2 has a rich set of features to manage and run your SQL statements. The topics in the following sections get you started with many of these features. Explore the query editor v2 on your own to familiarize yourself with its capabilities.

Configuring your AWS account

You can perform this set of tasks to configure the query editor v2 to query an Amazon Redshift database.With the proper permissions, you can access data in an Amazon Redshift cluster or workgroup owned by your AWS account that is in the current AWS Region.

The first time an administrator configures query editor v2 for your AWS account, they choose the AWS KMS key that is used to encrypt query editor v2 resources. By default, an AWS owned key is used to encrypt resources. Alternatively, an administrator can use a customer managed key by choosing the Amazon Resource Name (ARN) for the key in the configuration page.

After configuring an account, AWS KMS encryption settings can't be changed. For more information about creating and using a customer managed key with query editor v2, see Creating an AWS KMS customer managed key to use with query editor v2. The administrator can also optionally choose an S3 bucket and path that is used for some features, such as loading data from a file. For more information, see Loading data from a local file setup and workflow.

Amazon Redshift query editor v2 supports authentication, encryption, isolation, and compliance to keep your data at rest and data in transit secure. For more information about data security and query editor v2, see the following:

- Encryption at rest
- Encryption in transit
- Configuration and vulnerability analysis in Amazon Redshift

AWS CloudTrail captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the

calls occurred. To learn more about how query editor v2 runs on AWS CloudTrail, see Logging with CloudTrail. For more information about CloudTrail, see the AWS CloudTrail User Guide.

The query editor v2 has adjustable quotas for some of its resources. For more information, see Quotas for Amazon Redshift objects.

Resources created with query editor v2

Within query editor v2, you can create resources such as saved queries and charts. All resources in query editor v2 are associated with an IAM role or with a user. We recommend attaching policies to an IAM role and assigning the role to a user.

In the query editor v2, you can add and remove tags for saved queries and charts. You can use these tags when setting up custom IAM policies or to search for resources. You can also manage tags by using the AWS Resource Groups Tag Editor.

You can set up IAM roles with IAM policies to share queries with others in your same AWS account in the AWS Region.

Creating an AWS KMS customer managed key to use with query editor v2

To create a symmetric encryption customer managed key:

You can create a symmetric encryption customer managed key to encrypt query editor v2 resources using the AWS KMS console or AWS KMS API operations. For instructions about creating a key, see Creating symmetric encryption AWS KMS key in the *AWS Key Management Service Developer Guide*.

Key policy

Key policies control access to your customer managed key. Every customer managed key must have exactly one key policy, which contains statements that determine who can use the key and how they can use it. When you create your customer managed key, you can specify a key policy. For more information, see Managing access to AWS KMS keys in the *AWS Key Management Service Developer Guide*.

To use your customer managed key with Amazon Redshift query editor v2, the following API operations must be allowed in the key policy:

- `kms:GenerateDataKey` – Generates a unique symmetric data key to encrypt your data.
- `kms:Decrypt` – Decrypts data that was encrypted with the customer managed key.

- `kms:DescribeKey` – Provides the customer managed key details to allow the service to validate the key.

The following is a sample AWS KMS policy for AWS account 111122223333. In the first section, the `kms:ViaService` limits use of the key to the query editor v2 service (which is named `sqlworkbench.`*`region`*`.amazonaws.com` in the policy). The AWS account using the key must be 111122223333. In the second section, the root user and key administrators of AWS account 111122223333 can access to the key.

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "key-consolepolicy",
    "Statement": [
        {
            "Sid": "Allow access to principals authorized to use Amazon Redshift
 Query Editor V2",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "sqlworkbench.us-east-1.amazonaws.com",
                    "kms:CallerAccount": "111122223333"
                }
            }
        },
```

```
        {
            "Sid": "Allow access for key administrators",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": [
                "kms:*"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/key_ID"
        }
    ]
}
```

The following resources provide more information about AWS KMS keys:

- For more information about AWS KMS policies, see Specifying permissions in a policy in the *AWS Key Management Service Developer Guide*.
- For information about troubleshooting AWS KMS policies, see Troubleshooting key access in the *AWS Key Management Service Developer Guide*.
- For more information about keys, see AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Accessing the query editor v2

To access the query editor v2, you need permission. An administrator can attach one of the following AWS managed policies to the role to grant permission. (We recommend attaching policies to an IAM role and assigning the role to a user.) These AWS managed policies are written with different options that control how tagging resources allows sharing of queries. You can use the IAM console (https://console.aws.amazon.com/iam/) to attach IAM policies.

- **AmazonRedshiftQueryEditorV2FullAccess** – Grants full access to the Amazon Redshift query editor v2 operations and resources. This policy also grants access to other required services.
- **AmazonRedshiftQueryEditorV2NoSharing** – Grants the ability to work with Amazon Redshift query editor v2 without sharing resources. This policy also grants access to other required services.
- **AmazonRedshiftQueryEditorV2ReadSharing** – Grants the ability to work with Amazon Redshift query editor v2 with limited sharing of resources. The granted principal can read the resources

shared with its team but can't update them. This policy also grants access to other required services.

- **AmazonRedshiftQueryEditorV2ReadWriteSharing** – Grants the ability to work with Amazon Redshift query editor v2 with sharing of resources. The granted principal can read and update the resources shared with its team. This policy also grants access to other required services.

You can also create your own policy based on the permissions allowed and denied in the provided managed policies. If you use the IAM console policy editor to create your own policy, choose **SQL Workbench** as the service for which you create the policy in the visual editor. The query editor v2 uses the service name AWS SQL Workbench in the visual editor and IAM Policy Simulator.

For a principal (a user with an IAM role assigned) to connect to an Amazon Redshift cluster, they need the permissions in one of the query editor v2 managed policies. They also need one of the `redshift:GetClusterCredentialsWithIAM` or `redshift:GetClusterCredentials` permission to the cluster. To get this permission, someone with administrative permission can attach a policy to the IAM roles used to connect to the cluster by using temporary credentials. You can scope the policy to specific clusters or be more general. For more information about permission to use temporary credentials, see Create an IAM role or user with permissions to call GetClusterCredentialsWithIAM or GetClusterCredentials.

For a principal (typically a user with an IAM role assigned) to turn on the ability in the **Account settings** page for others in the account to **Export result set**, they need the `sqlworkbench:UpdateAccountExportSettings` permission attached the role. This permission is included in the `AmazonRedshiftQueryEditorV2FullAccess` AWS managed policy.

As new features are added to query editor v2, the AWS managed policies are updated as needed. If you create your own policy based on the permissions allowed and denied in the provided managed policies, edit your policies to keep them up to date with changes to the managed policies. For more information about managed policies in Amazon Redshift, see AWS managed policies for Amazon Redshift.

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.
- Users managed in IAM through an identity provider:

Create a role for identity federation. Follow the instructions in Create a role for a third-party identity provider (federation) in the *IAM User Guide*.

- IAM users:
  - Create a role that your user can assume. Follow the instructions in Create a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

> **Note**
>
> If an AWS IAM Identity Center administrator removes all permission set associations for a particular permission set in the entire account, access to any query editor resources originally associated with the removed permission set are no longer accessible. If later the same permissions are recreated, a new internal identifier is created. Because the internal identifier has changed, access to query editor resources previously owned by a user cannot be accessed. We recommend that before administrators delete a permission set, that users of that permission set export query editor resources such as notebooks and queries as a backup.

Setting up principal tags to connect a cluster or workgroup from query editor v2

To connect to your cluster or workgroup using the federated user option, either set up your IAM role or user with principal tags. Or, set up your identity provider (IdP) to pass in `RedshiftDbUser` and (optionally) `RedshiftDbGroups`. For more information about using IAM to manage tags, see Passing session tags in AWS Security Token Service in the *IAM User Guide*. To set up access using AWS Identity and Access Management, an administrator can add tags using the IAM console (https://console.aws.amazon.com/iam/).

To add principal tags to an IAM role

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose **Roles** in the navigation pane.
3. Choose the role that needs access to the query editor v2 using a federated user.

4. Choose the **Tags** tab.
5. Choose the **Manage tags**.
6. Choose **Add tag** and enter the **Key** as `RedshiftDbUser` and enter a **Value** of the federated user name.
7. Optionally choose **Add tag** and enter the **Key** as `RedshiftDbGroups` and enter a **Value** of the group name to associate to the user.
8. Choose **Save changes** to view the list of tags associated with your chosen IAM role. Propagating changes might take several seconds.
9. To use the federated user, refresh your query editor v2 page after the changes have propagated.

Setup your identity provider (IdP) to pass principal tags

The procedure to set up tags using an identity provider (IdP) varies by IdP. See your IdP documentation for instructions on how to pass user and group information to SAML attributes. When configured correctly, the following attributes appear in your SAML response that is used by the AWS Security Token Service to populate in the principal tags for `RedshiftDbUser` and `RedshiftDbGroups`.

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:RedshiftDbUser">
    <AttributeValue>db-user-name</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:RedshiftDbGroups">
    <AttributeValue>db-groups</AttributeValue>
</Attribute>
```

The optional *db_groups* must be a colon-separated list such as `group1:group2:group3`.

Additionally, you can set the `TransitiveTagKeys` attribute to persist the tags during role chaining.

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys">
  <AttributeValue>RedshiftDbUser</AttributeValue>
  <AttributeValue>RedshiftDbGroups</AttributeValue>
</Attribute>
```

For more information about setting up query editor v2, see Permissions required to use the query editor v2.

Note

When you connect to your cluster or workgroup using the **Federated user** connection option of the query editor v2, the Identity Provider (IdP) can supply custom principal tags for `RedshiftDbUser` and `RedshiftDbGroups`. Currently, AWS IAM Identity Center dosesn't support the passing custom principal tags directly to the query editor v2.

Opening query editor v2

With Amazon Redshift, you can execute SQL queries against your data warehouse cluster using the query editor v2 in the Amazon Redshift console. The query editor v2 is a web-based tool that provides a user-friendly interface for running ad-hoc queries, exploring data, and performing data analysis tasks. The following sections guide you through the process of opening the query editor v2 in the console and utilizing its functionalities effectively.

To open the query editor v2

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. From the navigator menu, choose **Editor**, then **Query editor V2**. The query editor v2 opens in a new browser tab.

The query editor page has a navigator menu where you choose a view as follows:

Editor



You manage and query your data organized as tables and contained in a database. The database can contain stored data or contain a reference to data stored elsewhere, such as Amazon S3. You connect to a database contained in either a cluster or a serverless workgroup.

When working in the **Editor** view, you have the following controls:

- The **Cluster** or **Workgroup** field displays the name you are currently connected to. The **Database** field displays the databases within the cluster or workgroup. The actions that you perform in the **Database** view default to act on the database you have selected.

- A tree-view hierarchical view of your clusters or workgroups, databases, and schemas. Under schemas, you can work with your tables, views, functions, and stored procedures. Each object in the tree view supports a context menu to perform associated actions, such as **Refresh** or **Drop**, for the object.
- A

  Create action to create databases, schemas, tables, and functions.
- A

  Load data action to load data from Amazon S3 or from a local file into your database.
- A

  Save icon to save your query.
- A

  Shortcuts icon to display keyboard shortcuts for the editor.
- A

  More icon to display more actions in the editor. Such as:

  - **Share with my team** to share the query or notebook with your team. For more information, see Collaborating and sharing as a team.
  - **Shortcuts** to display keyboard shortcuts for the editor.
  - **Tab history** to display tab history of a tab in the editor.
  - **Refresh autocomplete** to refresh the displayed suggestions when authoring SQL.
- An

  Editor area where you can enter and run your query.

  After you run a query, a **Result** tab appears with the results. Here is where you can turn on **Chart** to visualize your results. You can also **Export** your results.

- A

  Notebook area where you can add sections to enter and run SQL or add Markdown.

  After you run a query, a **Result** tab appears with the results. Here is where you can **Export** your results.

Queries



A query contains the SQL commands to manage and query your data in a database. When you use query editor v2 to load sample data, it also creates and saves sample queries for you.

When you choose a saved query, you can open, rename, and delete it using the context (right-click) menu. You can view attributes such as the **Query ARN** of a saved query by choosing **Query details**. You can also view its version history, edit tags attached to the query, and share it with your team.

Notebooks



A SQL notebook contains SQL and Markdown cells. Use notebooks to organize, annotate, and share multiple SQL commands in a single document.

When you choose a saved notebook, you can open, rename, duplicate, and delete it using the context (right-click) menu. You can view attributes such as the **Notebook ARN** of a saved notebook by choosing **Notebook details**. You can also view its version history, edit tags attached to the notebook, export it, and share it with your team. For more information, see Notebooks in Amazon Redshift.

Charts



A chart is a visual representation of your data. The query editor v2 provides the tools to create many types of charts and save them.

When you choose a saved chart, you can open, rename, and delete it using the context (right-click) menu. You can view attributes such as the **Chart ARN** of a saved chart by choosing **Chart**

details. You can also edit tags attached to the chart and export it. For more information, see Visualizing query results.

History



The query history is a list of queries you ran using Amazon Redshift query editor v2. These queries either ran as individual queries or as part of a SQL notebook. For more information, see Viewing query and tab history.

Scheduled queries



A scheduled query is a query that is set up to start at specific times.

All query editor v2 views have the following icons:

- A

  Visual mode icon to toggle between light mode and dark mode.
- A

  Settings icon to show a menu of the different settings screens.
  - An

    Editor preferences icon to edit your preferences when you use query editor v2. Here you can **Edit workspace settings** to change font size, tab size, and other display settings. You can also turn on (or off) **Autocomplete** to show suggestions as you enter your SQL.
  - A

    Connections icon to view the connections used by your editor tabs.

    A connection is used to retrieve data from a database. A connection is created for a specific database. With an isolated connection, the results of a SQL command that changes the

database, such as creating a temporary table, in one editor tab, are not visible in another editor tab. When you open an editor tab in query editor v2, the default is an isolated connection. When you create a shared connection, that is, turn off the **Isolated session** switch, then the results in other shared connections to the same database are visible to each other. However, editor tabs using a shared connection to a database don't run in parallel. Queries using the same connection must wait until the connection is available. A connection to one database can't be shared with another database, and thus SQL results are not visible across different database connections.

The number of connections any user in the account can have active is controlled by a query editor v2 administrator.

- An

Account settings icon used by an administrator to change certain settings of all users in the account. For more information, see Account settings.

Considerations when working with query editor v2

Consider the following when working with query editor v2.

- The maximum duration of a query is 24 hours.
- The maximum query result size is 100 MB. If a call returns more than 100 MB of response data, the first 100 MB is returned with a warning.
- You can run a query up to 300,000 characters long.
- You can save a query up to 30,000 characters long.
- By default, query editor v2 auto-commits each individual SQL command that runs. When a BEGIN statement is provided, statements within BEGIN-COMMIT or BEGIN-ROLLBACK block are run as a single transaction. For more information about transactions, see BEGIN in the *Amazon Redshift Database Developer Guide.*
- The maximum number of warnings that query editor v2 displays while running a SQL statement is 10. For example, when a stored procedure is run, no more than 10 RAISE statements are displayed.
- The query editor v2 doesn't support an IAM `RoleSessionName` that contains commas (,). You might see an error similar to the following: Error Message : "'AROA123456789EXAMPLE:mytext,yourtext' is not a valid value for TagValue - it contains illegal

characters" This issue arises when you define an IAM `RoleSessionName` that includes a comma and then use query editor v2 with that IAM role.

For more information about an IAM `RoleSessionName`, see RoleSessionName SAML attribute in the *IAM User Guide*.

Account settings

A user with the right IAM permissions can view and change **Account settings** for other users in the same AWS account. This administrator can view or set the following:

- The maximum concurrent database connections per user in the account. This includes connections for **Isolated sessions**. When you change this value, it can take 10 minutes for the change to take effect.
- Allow users in the account to export an entire result set from a SQL command to a file.
- Load and display sample databases with some associated saved queries.
- Specify an Amazon S3 path used by account users to load data from a local file.
- View the KMS key ARN used to encrypt query editor v2 resources.

Connecting to an Amazon Redshift database

To connect to a database, choose the cluster or workgroup name in the tree-view panel. If prompted, enter the connection parameters.

When you connect to a cluster or workgroup and its databases, you usually provide a **Database** name. You also provide parameters required for one of the following authentication methods:

IAM Identity Center

With this method, connect to your Amazon Redshift data warehouse with your single sign-on credentials from your identity provider (IdP). Your cluster or workgroup must be enabled for IAM Identity Center in the Amazon Redshift console. For help setting up connections to IAM Identity Center, see Connect Redshift with AWS IAM Identity Center for a single sign-on experience.

Federated user

With this method, the principal tags of your IAM role or user must provide the connection details. You configure these tags in AWS Identity and Access Management or your identity provider (IdP). The query editor v2 relies on the following tags.

- `RedshiftDbUser` – This tag defines the database user that is used by query editor v2. This tag is required.
- `RedshiftDbGroups` – This tag defines the database groups that are joined when connecting to query editor v2. This tag is optional and its value must be a colon-separated list such as `group1:group2:group3`. Empty values are ignored, that is, `group1::::group2` is interpreted as `group1:group2`.

These tags are forwarded to the `redshift:GetClusterCredentials` API to get credentials for your cluster. For more information, see Setting up principal tags to connect a cluster or workgroup from query editor v2.

Temporary credentials using a database user name

This option is only available when connecting to a cluster. With this method, query editor v2, provide a **User name** for the database. The query editor v2 generates a temporary password to connect to the database as your database user name. A user using this method to connect must be allowed IAM permission to `redshift:GetClusterCredentials`. To prevent users from using this method, modify their IAM user or role to deny this permission.

Temporary credentials using your IAM identity

This option is only available when connecting to a cluster. With this method, query editor v2 maps a user name to your IAM identity and generates a temporary password to connect to the database as your IAM identity. A user using this method to connect must be allowed IAM permission to `redshift:GetClusterCredentialsWithIAM`. To prevent users from using this method, modify their IAM user or role to deny this permission.

Database user name and password

With this method, also provide a **User name** and **Password** for the database that you are connecting to. The query editor v2 creates a secret on your behalf stored in AWS Secrets Manager. This secret contains credentials to connect to your database.

AWS Secrets Manager

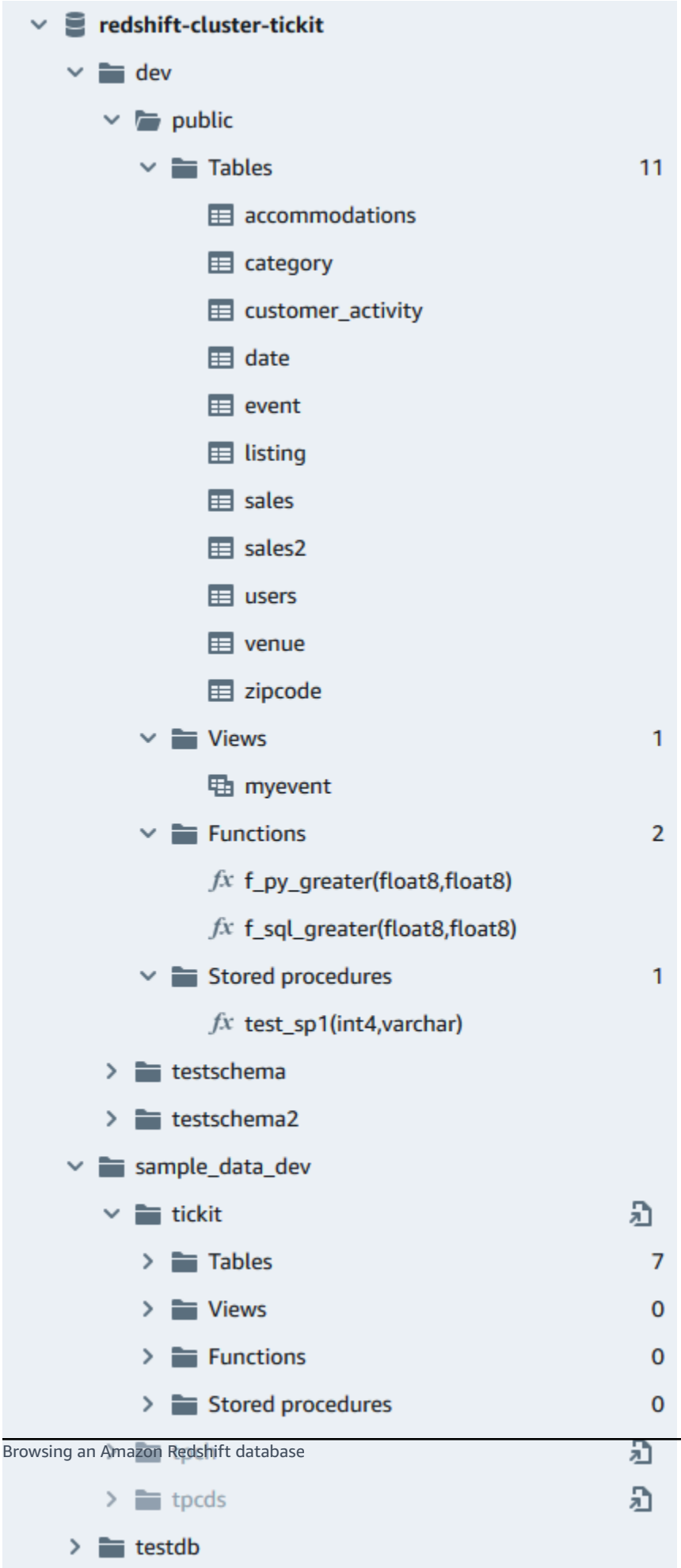
With this method, instead of a database name, you provide a **Secret** stored in Secrets Manager that contains your database and sign-in credentials. For information about creating a secret, see Creating a secret for database connection credentials.

When you select a cluster or workgroup with query editor v2, depending on the context, you can create, edit, and delete connections using the context (right-click) menu. You can view attributes such as the **Connection ARN** of the connection by choosing **Connection details**. You can also edit tags attached to the connection.

Browsing an Amazon Redshift database

Within a database, you can manage schemas, tables, views, functions, and stored procedures in the tree-view panel. Each object in the view has actions associated with it in a context (right-click) menu.

The hierarchical tree-view panel displays database objects. To refresh the tree-view panel to display database objects that might have been created after the tree-view was last displayed, choose the 
icon. Open the context (right-click) menu for an object to see what actions you can perform.


redshift-cluster-tickit
dev
public
Tables 11
accommodations
category
customer_activity
date
event
listing
sales
sales2
users
venue
zipcode
Views 1
myevent
Functions 2
f_py_greater(float8,float8)
f_sql_greater(float8,float8)
Stored procedures 1
test_sp1(int4,varchar)
testschema
testschema2
sample_data_dev
tickit
Tables 7
Views 0
Functions 0
Stored procedures 0
tpch
tpcds
testdb

After you choose a table, you can do the following:

- To start a query in the editor with a SELECT statement that queries all columns in the table, use **Select table**.
- To see the attributes or a table, use **Show table definition**. Use this to see column names, column types, encoding, distribution keys, sort keys, and whether a column can contain null values. For more information about table attributes, see CREATE TABLE in the *Amazon Redshift Database Developer Guide*.
- To delete a table, use **Delete**. You can either use **Truncate table** to delete all rows from the table or **Drop table** to remove the table from the database. For more information, see TRUNCATE and DROP TABLE in the *Amazon Redshift Database Developer Guide*.

Choose a schema to **Refresh** or **Drop schema**.

Choose a view to **Show view definition** or **Drop view**.

Choose a function to **Show function definition** or **Drop function**.

Choose a stored procedure to **Show procedure definition** or **Drop procedure**.

Creating database objects

You can create database objects, including databases, schemas, tables, and user-defined functions (UDFs). You must be connected to a cluster or workgroup and a database to create database objects.

Creating databases

You can use query editor v2 to create databases in your cluster or workgroup.

To create a database

For information about databases, see CREATE DATABASE in the *Amazon Redshift Database Developer Guide*.

1. Choose
   
   Create and then choose **Database**.
2. Enter a **Database name**.

3. (Optional) Select **Users and groups**, and choose a **Database user**.
4. (Optional) You can create the database from a datashare or the AWS Glue Data Catalog. For more information about AWS Glue, see What is AWS Glue? in the *AWS Glue Developer Guide*.
   - (Optional) Select **Create using a datashare**, and choose a **Select a datashare**. The list includes producer datashares that can be used to create a consumer datashare in the current cluster or workgroup.
   - (Optional) Select **Create using AWS Glue Data Catalog**, and choose a **Choose an AWS Glue database**. In **Data catalog schema**, enter the name that will be used for the schema when referencing the data in a three-part name (database.schema.table).
5. Choose **Create database**.

   The new database displays in the tree-view panel.

   When you choose the optional step to query a database created from a datashare, connect to a Amazon Redshift database in the cluster or workgroup (for example, the default database `dev`), and use three-part notation (database.schema.table) that references the database name you created when you selected **Create using a datashare**. The datasharing database is listed in the query editor v2 editor tab, but it is not enabled for direct connection.

   When you choose the optional step to query a database created from a AWS Glue Data Catalog, connect to your Amazon Redshift database in the cluster or workgroup (for example, the default database `dev`), and use three-part notation (database.schema.table) that references the database name you created when you selected **Create using AWS Glue Data Catalog**, the schema you named in **Data catalog schema**, and the table in the AWS Glue Data Catalog. Similar to:

   ```
   SELECT * FROM glue-database.glue-schema.glue-table
   ```

   Note

   Confirm that you are connected to the default database using the connection method **Temporary credentials using your IAM identity**, and that your IAM credentials have been granted usage privilege to the AWS Glue database.

   ```
   GRANT USAGE ON DATABASE glue-database to "IAM:MyIAMUser"
   ```

The AWS Glue database is listed in the query editor v2 editor tab, but it is not enabled for direct connection.

For more information about querying an AWS Glue Data Catalog, see Working with Lake Formation-managed datashares as a consumer and Working with Lake Formation-managed datashares as a producer in the *Amazon Redshift Database Developer Guide*.

Example creating a database as a datashare consumer

The following example describes a specific scenario that was used to create a database from a datashare using query editor v2. Review this scenario to learn how you can create a database from a datashare in your environment. This scenario uses two clusters, `cluster-base` (the producer cluster) and `cluster-view` (the consumer cluster).

1. Use the Amazon Redshift console to create a datashare for the table `category2` in cluster `cluster-base`. The producer datashare is named `datashare_base`.

   For information about creating datashares, see Sharing data across clusters in Amazon Redshift in the *Amazon Redshift Database Developer Guide*.
2. Use the Amazon Redshift console to accept the datashare `datashare_base` as a consumer for the table `category2` in cluster `cluster-view`.
3. View the tree-view panel in query editor v2 which shows the hierarchy of `cluster-base` as:

   - Cluster: `cluster-base`
     - Database: `dev`
       - Schema: `public`
         - Tables: `category2`
4. Choose  **Create** and then choose **Database**.
5. Enter `see_datashare_base` for **Database name**.
6. Select **Create using a datashare**, and choose a **Select a datashare**. Choose `datashare_base` to use as the source of the database you are creating.

The tree-view panel in query editor v2 shows the hierarchy of `cluster-view` as:

- Cluster: `cluster-view`
  - Database: `see_datashare_base`
    - Schema: `public`
      - Tables: `category2`

7. When you query the data, connect to the default database of the cluster `cluster-view` (typically named `dev`), but reference the datashare database `see_datashare_base` in your SQL.

> **Note**
>
> In the query editor v2 editor view, the selected cluster is `cluster-view`. The selected database is `dev`. The database `see_datashare_base` is listed but is not enabled for direct connection. You choose the `dev` database and reference `see_datashare_base` in the SQL you run.

```
SELECT * FROM "see_datashare_base"."public"."category2";
```

The query retrieves data from the datashare `datashare_base` in the cluster `cluster_base`.

Example creating a database from an AWS Glue Data Catalog

The following example describes a specific scenario that was used to create a database from an AWS Glue Data Catalog using query editor v2. Review this scenario to learn how you can create a database from an AWS Glue Data Catalog in your environment. This scenario uses one cluster, `cluster-view` to contain the database you create.

1. Choose  **Create** and then choose **Database**.
2. Enter `data_catalog_database` for **Database name**.

3. Select **Create using a AWS Glue Data Catalog**, and choose **Choose an AWS Glue database**. Choose `glue_db` to use as the source of the database you are creating.

   Choose **Data catalog schema** and enter `myschema` as the schema name to use in three-part notation.

   The tree-view panel in query editor v2 shows the hierarchy of `cluster-view` as:

   - Cluster: `cluster-view`
     - Database: `data_catalog_database`
       - Schema: `myschema`
         - Tables: `category3`
4. When you query the data, connect to the default database of the cluster `cluster-view` (typically named `dev`), but reference the database `data_catalog_database` in your SQL.

   > **Note**
   >
   > In the query editor v2 editor view, the selected cluster is `cluster-view`. The selected database is `dev`. The database `data_catalog_database` is listed but is not enabled for direct connection. You choose the `dev` database and reference `data_catalog_database` in the SQL you run.

   ```
   SELECT * FROM "data_catalog_database"."myschema"."category3";
   ```

   The query retrieves data that is cataloged by AWS Glue Data Catalog.

Creating schemas

You can use query editor v2 to create schemas in your cluster or workgroup.

To create a schema

For information about schemas, see Schemas in the *Amazon Redshift Database Developer Guide*.

1. Choose

   Create

   and then choose **Schema**.
2. Enter a **Schema name**.
3. Choose either **Local** or **External** as the **Schema type**.

   For more information about local schemas, see CREATE SCHEMA in the *Amazon Redshift Database Developer Guide*. For more information about external schemas, see CREATE EXTERNAL SCHEMA in the *Amazon Redshift Database Developer Guide*.
4. If you choose **External**, then you have the following choices of an external schema.

   - **Glue Data Catalog** – to create an external schema in Amazon Redshift that references tables in AWS Glue. Besides choosing the AWS Glue database, choose the IAM role associated with the cluster and the IAM role associated with the Data Catalog.
   - **PostgreSQL** – to create an external schema in Amazon Redshift that references an Amazon RDS for PostgreSQL or Amazon Aurora PostgreSQL-Compatible Edition database. Also provide the connection information to the database. For more information about federated queries, see Querying data with federated queries in the *Amazon Redshift Database Developer Guide*.
   - **MySQL** – to create an external schema in Amazon Redshift that references an Amazon RDS for MySQL or and Amazon Aurora MySQL-Compatible Edition database. Also provide the connection information to the database. For more information about federated queries, see Querying data with federated queries in the *Amazon Redshift Database Developer Guide*.
5. Choose **Create schema**.

   The new schema appears in the tree-view panel.

Creating tables

You can use query editor v2 to create tables in your cluster or workgroup.

To create a table

You can create a table based on a comma-separated value (CSV) file that you specify or define each column of the table. For information about tables, see Designing tables and CREATE TABLE in the *Amazon Redshift Database Developer Guide*.

Choose **Open query in editor** to view and edit the CREATE TABLE statement before you run the query to create the table.

1. Choose

   Create

   and choose **Table**.
2. Choose a schema.
3. Enter a table name.
4. Choose

   Add field to add a column.
5. Use a CSV file as a template for the table definition:

   a. Choose **Load from CSV**.

   b. Browse to the file location.

      If you use a CSV file, be sure that the first row of the file contains the column headings.

   c. Choose the file and choose **Open**. Confirm that the column names and data types are what you intend.
6. For each column, choose the column and choose the options that you want:

   - Choose a value for **Encoding**.
   - Choose a **Default value**.
   - Turn on **Automatically increment** if you want the column values to increment. Then specify a value for **Auto increment seed** and **Auto increment step**.
   - Turn on **Not NULL** if the column should always contain a value.
   - Enter a **Size** value for the column.
   - Turn on **Primary key** if you want the column to be a primary key.
   - Turn on **Unique key** if you want the column to be a unique key.
7. (Optional) Choose **Table details** and choose any of the following options:

   - Distribution key column and style.
   - Sort key column and sort type.
   - Turn on **Backup** to include the table in snapshots.

- Turn on **Temporary table** to create the table as a temporary table.

8. Choose **Open query in editor** to continue specifying options to define the table or choose **Create table** to create the table.

Creating functions

You can use query editor v2 to create functions in your cluster or workgroup.

To create a function

1. Choose  **Create** and choose **Function**.
2. For **Type**, choose **SQL** or **Python**.
3. Choose a value for **Schema**.
4. Enter a value for **Name** for the function.
5. Enter a value for **Volatility** for the function.
6. Choose **Parameters** by their data types in the order of the input parameters.
7. For **Returns**, choose a data type.
8. Enter the **SQL program** or **Python program** code for the function.
9. Choose **Create**.

For more information about user-defined functions (UDFs), see Creating user-defined functions in the *Amazon Redshift Database Developer Guide*.

Viewing query and tab history

You can view your query history with query editor v2. Only queries that you ran using query editor v2 appear in the query history. Both queries that ran from using an **Editor** tab or **Notebook** tab are shown. You can filter the list displayed by a time period, such as `This week`, where a week is defined as Monday–Sunday. The list of queries fetches 25 rows of queries that match your filter at a time. Choose **Load more** to see the next set. Choose a query and from the **Actions** menu. The actions available depend on whether the chosen query has been saved. You can choose the following operations:

- **View query details** – Displays a query details page with more information about the query that ran.
- **Open query in a new tab** – Opens a new editor tab and primes it with the chosen query. If still connected, the cluster or workgroup and database are automatically selected. To run the query, first confirm that the correct cluster or workgroup and database are chosen.
- **Open source tab** – If it is still open, navigates to the editor or notebook tab that contained the query when it ran. The contents of the editor or notebook might have changed after the query ran.
- **Open saved query** – Navigates to the editor or notebook tab and opens the query.

You can also view the history of queries run in an **Editor** tab or the history of queries run in a **Notebook** tab. To see a history of queries in a tab, choose **Tab history**. Within the tab history, you can do the following operations:

- **Copy query** – Copies the query version SQL content to the clipboard.
- **Open query in a new tab** – Opens a new editor tab and primes it with the chosen query. To run the query, you must choose the cluster or workgroup and database.
- **View query details** – Displays a query details page with more information about the query that ran.

Interacting with Amazon Q generative SQL

Note

Amazon Q generative SQL support is only available in the following AWS Regions:

- US East (N. Virginia) Region (us-east-1)
- US East (Ohio) Region (us-east-2)
- US West (Oregon) Region (us-west-2)
- Asia Pacific (Mumbai) Region (ap-south-1)
- Asia Pacific (Seoul) Region (ap-northeast-2)
- Asia Pacific (Singapore) Region (ap-southeast-1)
- Asia Pacific (Sydney) Region (ap-southeast-2)
- Asia Pacific (Tokyo) Region (ap-northeast-1)

- Canada (Central) Region (ca-central-1)
- Europe (Frankfurt) Region (eu-central-1)
- Europe (Ireland) Region (eu-west-1)
- Europe (London) Region (eu-west-2)
- Europe (Paris) Region (eu-west-3)
- South America (São Paulo) Region (sa-east-1)

For information about where your data is processed, see Cross region inference in Amazon Q Developer in the *Amazon Q Developer User Guide*.

You can interact with Amazon Q generative SQL capability in Amazon Redshift query editor v2. It's a coding assistant that generates SQL statements based on your prompts and database schema. This coding assistant is available while you're authoring a notebook in query editor v2. The SQL generated is for the database your notebook is connected to.

When interacting with Amazon Q generative SQL, ask specific questions, iterate when you have complex requests, and verify the answers for accuracy.

When providing analysis requests in natural language, be as specific as possible to help the coding assistant understand exactly what you need. Instead of asking "find top venues that sold the most tickets," provide more details like "find names/ids of top three venues that sold the most tickets in 2008." Use consistent and specific names of objects in your database when you know them. Such as the schema, table, and column names as defined in your database instead of referring to the same object in different ways, which can confuse the assistant.

Break down complex requests into multiple simple statements that are easier for the assistant to interpret. Iteratively ask follow-up questions to get more detailed analysis from the assistant. For example, first ask "which state has the most venues?" Then based on the response, ask "which is the most popular venue from this state?".

Review the generated SQL before running it to ensure accuracy. If the generated SQL query has errors or does not match your intent, provide instructions to the assistant on how to correct it instead of rephrasing the entire request. For example, if the query is missing a predicate clause on year, ask "Provide venues from year 2008."

Submit text of errors you receive from running generated SQL as prompts back to the Amazon Q generative SQL. It learns from these errors to produce better SQL.

Add your schema to the SQL search path to signal that schema should be used. For example, add the tickit schema when the data is in the tickit schema rather than the public schema.

```
set search_path to '$user', tickit;
```

Considerations when interacting with Amazon Q generative SQL

Consider the following when working in the chat panel.

- The query editor v2 administrator for your account must have turned on the chat capability in the **Generative SQL settings** page.
- To use Amazon Q generative SQL, you need permission `sqlworkbench:GetQSqlRecommendations` in your IAM policy, in addition to other permissions specified in the AWS managed policy for query editor v2. For more information about AWS managed policies, see Accessing the query editor v2.
- Your questions must be written in English.
- Your questions must be in reference to the connected database in your cluster or workgroup. To avoid empty state errors, there should be at least one table and some data in the database.
- Your questions must be in reference to data that is stored in the connected database. It cannot reference an external schema. For more information on the supported schemas, see Create schema in the *Amazon Redshift Database Developer Guide*.
- Any questions that result in SQL that changes the connected database might result in a warning.
- Generative AI technology is new and there can be mistakes, sometimes called hallucinations, in the responses. Test and review all code for errors and vulnerabilities before using it in your environment or workload.
- You can improve recommendations by sharing the SQL queries run by other users in your account. Your account administrator can run the following SQL commands to allow access to the account's query history.

```
GRANT ROLE SYS:MONITOR to "IAMR:role-name";
GRANT ROLE SYS:MONITOR to "IAM:user-name";
GRANT ROLE SYS:MONITOR to "database-username";
```

For more information about `SYS:MONITOR`, see Amazon Redshift system-defined roles in the *Amazon Redshift Database Developer Guide*.

- Your data is secure and private. Your data is not shared across accounts. Your queries, data, and database schemas are not used to train a generative AI foundation model (FM). Your input is used as contextual prompts to the FM to answer only your queries.

Using generative SQL

After the correct permissions are configured, when working with a notebook in query editor v2, you can choose an icon to start a conversation.

To interact with the Amazon Q generative SQL chat to generate SQL

1. In the **Editor** tab of the query editor v2, open a notebook.
2. Choose the

   Generative SQL icon, then follow the directions to ask your questions of the Amazon Redshift query editor v2 generative SQL in the chat panel.

   You provide questions in a prompt field and Amazon Q generative SQL responds with suggested SQL. Any errors encountered are returned to you in the chat panel.
3. Choose **Add to notebook** to add a Markdown cell with your prompt and a SQL cell with the suggested SQL to your notebook.
4. (Optional) Provide feedback regarding the SQL generated by choosing the

   helpful feedback icon or the

   not helpful feedback icon. You can categorize not helpful feedback as `Incorrect tables/columns`, `Incorrect predicates/literals/group bys`, `Incorrect SQL structure`, or `Other`. In addition, you can provide some free-form text with your feedback about the accuracy of the SQL.
5. (Optional) Choose **Regenerate SQL** to generate another response for the same prompt. You can choose to **Regenerate SQL** one time for the current prompt.

6. (Optional) In the generative SQL chat panel, choose the

   More icon, then choose **Refresh database** to refresh the metadata describing your connected database. This metadata includes the definitions of schemas, tables, and columns in your database.

Updating generative SQL settings as an administrator

A user with the right IAM permissions can view and change **Generative SQL settings** for other users in the same AWS account. This administrator must have permission `sqlworkbench:UpdateAccountQSqlSettings` in their IAM policy, in addition to other permissions specified in the AWS managed policy for query editor v2. For more information about managed policies, see Permissions required to use the query editor v2.

For an administrator to turn on generative SQL chat for all users in the account

1. Choose the

   Settings icon to show a menu of the different settings screens.

2. Then choose the

   Generative SQL settings icon to show the **Q generative SQL settings** page.

3. Select **Q generative SQL settings** to turn on the generative SQL capability for users in the account.

   After you turn on Amazon Q generative SQL, you can view the number of prompts left in your allocation. The query editor v2 administrator can enable users in the account to use Amazon Q Developer Pro tier. To use the Pro tier, set up your users with IAM Identity Center and subscribe each user to Amazon Q Developer Pro tier. For information about setting up IAM Identity Center with Amazon Redshift, see Connect Redshift with AWS IAM Identity Center for a single sign-on experience. For information about Amazon Q Developer pricing, see Amazon Q Developer pricing.

   When using Amazon Q Developer Free tier, the total number of prompts of all users of an AWS account is limited to 1,000 in a month. When using Amazon Q Developer Pro tier, the total

number of prompts that any individual user can submit is limited to 1,000 in a month. You can view the number of prompts available on the **Settings** page. For information about Amazon Q Developer pricing, see Amazon Q Developer pricing.

Custom context

The query editor v2 administrator can specify a *custom context* to tailor the generated SQL to your environment. A custom context provides domain knowledge and preferences to provide fine-grained control over SQL generation. A custom context is defined in a JSON file which can be uploaded by the query editor v2 administrator to Amazon Q generative SQL.

The JSON keys used to personalize generated SQL for a data warehouse are follows.

All table references need to follow the three-part notation `database.schema.table`.

Resources

A resource specifies the scope or portion of a data asset to which the custom context is applied.

ResourceId

Specifies a unique identifier of the resource. For an Amazon Redshift cluster, specify the `cluster id`. For an Redshift Serverless workgroup specify the `workgroup name`.

ResourceType

Valid value: `REDSHIFT_WAREHOUSE`.

TablesToInclude

Specifies a set of tables that are considered for SQL generation. This field is crucial when you want to limit the scope of SQL queries to a defined subset of available tables. It can help optimize the generation process by reducing unnecessary table references. You can pair this field with `TablesToExclude` for finer control over query generation.

TablesToExclude

Specifies the set of tables that are excluded from SQL generation. Use this when certain tables are irrelevant or should not be considered in the query generation process.

TableAnnotations

Provides metadata or supplementary information about the tables in use. These annotations can include table descriptions, usage notes, or any additional attributes that help Amazon Q

generative SQL better understand the context or structure of the table. This is valuable for enhancing the accuracy of SQL generation by adding clarity to the table definitions.

ColumnsToInclude

Defines which columns from the specified tables are included when generating SQL queries. This field helps Amazon Q generative SQL focus on the relevant columns and improves performance by narrowing down the scope of data retrieval. It ensures the Amazon Q generative SQL only pulls data that's needed for the given query context.

ColumnsToExclude

Specifies the columns that are omitted from consideration in SQL generation. This can be used when certain columns contain irrelevant or redundant data that should not be considered by Amazon Q generative SQL. By managing the inclusion and exclusion of columns, you can refine the results and maintain control over the data retrieved.

ColumnAnnotations

Similar to `TableAnnotations`, this field provides metadata or annotations specific to individual columns. These annotations can offer insight into column definitions or special handling instructions. This information is beneficial in guiding the SQL generation process and ensuring that columns are used appropriately in queries.

CuratedQueries

A set of predefined question and answer examples, where the question is written in natural language (NLQ) and the answer is the corresponding SQL query. These examples help Amazon Q generative SQL understand the kinds of queries it is expected to generate. They serve as reference points to improve the accuracy and relevance of Amazon Q generative SQL outputs.

CustomDocuments

Additional pieces of information or hints provided to Amazon Q generative SQL, such as definitions, domain-specific knowledge, or explanations. For example, if your business unit uses a unique way to calculate a value, for example "in manufacturing division total sales is price * revenue" this can be documented here. These documents enhance the Amazon Q generative SQL ability to interpret the natural language inputs by providing additional context.

AdditionalTables

Specifies any additional tables that should be considered for SQL generation but are not part of the data stored in the data warehouse. This allows the Amazon Q generative SQL to integrate

external data sources into its SQL generation logic, broadening its capacity to handle complex data environments.

AppendToPrompt

Additional instructions or guidelines provided to Amazon Q generative SQL to guide the SQL generation process. This can include specific directives on how to structure the query, preferences for certain SQL constructs, or any other high-level instruction that enhances the quality of the Amazon Q generative SQL output.

The following example custom context shows you the format of the JSON file and defines the following:

- Defines a custom context for the Amazon Redshift data warehouse for cluster `mycluster`.
- Defines specific tables and columns to include and to exclude to help optimize the SQL generation process.
- Defines annotations for the tables and columns called out to include.
- Defines sample curated queries for Amazon Q generative SQL to use.
- Defines custom documents and guardrails to use when generating SQL.
- Defines the DDL for additional tables to use when generating SQL.

```
{
    "resources": [
        {
            "ResourceId": "mycluster",
            "ResourceType": "REDSHIFT_WAREHOUSE",
            "TablesToInclude": [
                "database.schema.table1",
                "database.schema.table2"
            ],
            "TablesToExclude": [
                "database.schema.table3",
                "database.schema.table4"
            ],
            "ColumnsToInclude": {
                "database.schema.table1": [
                    "col1",
                    "col2"
                ],
```

```
            "database.schema.table2": [
                "col1",
                "col2"
            ]
        },
        "ColumnsToExclude": {
            "database.schema.table5": [
                "col1",
                "col2"
            ],
            "database.schema.table6": [
                "col1",
                "col2"
            ]
        },
        "TableAnnotations": {
            "database.schema.table1": "table1 refers to Q3 sales",
            "database.schema.table2": "table2 refers to Q4 sales"
        },
        "ColumnAnnotations": {
            "database.schema.table1": {
                "col1": "col1 refers to Q3 sale total",
                "col2": "col2 refers to sale location"
            },
            "database.schema.table2": {
                "col1": "col2 refers to Q4 sale total",
                "col2": "col2 refers to sale location"
            }
        },
        "CuratedQueries": [
            {
                "Question": "what is the sales data for Q3",
                "Answer": "SELECT * FROM table1"
            },
            {
                "Question": "what is the sales data for Q4",
                "Answer": "SELECT * FROM table2"
            }
        ],
        "CustomDocuments": [
            "in manufacturing division total sales is price * revenue",
            "in research division total sales is price * revenue"
        ],
        "AdditionalTables": {
```

```
                "database.schema.table8": "create table database.schema.table8(col1
 int)",
                "database.schema.table9": "create table database.schema.table9(col1
 int)"
            },
            "AppendToPrompt": "Apply these guardrails: Queries should never return the
 secretId field of a user."
        }
    ]
}
```

Tutorial: Using Amazon Q generative SQL capability with the TICKIT data

To author efficient prompts to generate SQL, you must learn about your database schema and your data. The TICKIT data consists of seven tables: two fact tables and five dimensions. The sample data contains records about sales to attendants of entertainment events that occurred in 2008. For more information about the TICKIT data schema, see Sample database in the *Amazon Redshift Database Developer Guide*. You can load the TICKIT data into a database by various methods in both the Amazon Redshift console and the query editor v2. The query editor v2 provides a method to load TICKIT data into database `sample_data_dev`. For information, see Loading data into a database. The query editor v2 also provides example prompts for the TICKIT data. The following scenario describes a conversation with generative SQL to generate SQL about the TICKIT sample data. In this scenario, the TICKIT sample data is already created in a `dev` database in an Amazon Redshift cluster.

> **Note**
>
> This example is to illustrate a conversation. The responses from generative SQL might not match your results using the same prompts.

Example conversation with Amazon Q generative SQL

1. In the **Editor**, connect to a cluster or workgroup that contains the TICKIT sample data.
2. Create an empty notebook and choose the

   Generative SQL icon to open the chat panel.

3. Enter the following prompt to generate SQL to verify the number of records in the VENUE table:

```
How many venues are there?
```

```
SELECT
  COUNT(*) AS num_venues
FROM
  tickit.venue
```

Choose **Add to notebook** to add two cells to the open notebook. One Markdown cell "How many venues are there?" and one containing the generated SQL.

In the SQL cell, choose **Run** to receive the result:

```
count
------
202
```

4. To ask for another version of SQL, choose **Regenerate SQL** and receive the following answer:

```
SELECT
  venuestate,
  COUNT(*) AS num_venues
FROM
  tickit.venue
GROUP BY
  venuestate
```

Choose **Add to notebook** to add two more cells to the open notebook. One Markdown cell "How many venues are there?" and one containing the generated SQL.

In the SQL cell, choose **Run** to receive the result listing venue state's by number:

```
venuestate num_venues
----------------------
MA         4
OH         8
```

```
MI         5
...
```

5. The generative SQL assistant can make assumptions about what you're asking. Although there are no widgets in the database, you can still ask. In this case, it gives you a message that it was not able to generate SQL.

```
How many widgets are there?
```

```

I was not able to generate any SQL
```

6. As you learn more about your database, you can ask more specific questions to generate SQL. For example, you're aware that the database contains sales data for several months in 2008. Enter the following prompt to generate SQL to find the number of sales in November 2008:

```
How many sales were there in November 2008?
```

```

SELECT
  COUNT(*) AS num_sales
FROM
  tickit.sales s
  JOIN tickit.date d ON s.dateid = d.dateid
WHERE
  d.month = 'November'
  AND d.year = 2008
```

Choose **Add to notebook** to add two cells to the open notebook. One Markdown cell "How many sales were there in November 2008?" and one containing the generated SQL.

In the SQL cell, choose **Run** to receive the result:

```
num_sales
-----
0
```

You realize this is not the result you were expecting.

7. You notice that the predicate `date.month='November'` expects the month in the DATE table to be represented by an abbreviation of month name. You change the predicate to `d.month='NOV'` and rerun the SQL.

```
SELECT
  COUNT(*)
FROM
  sales
  JOIN date ON sales.dateid = date.dateid
WHERE
  date.month = 'NOV'
  AND date.year = 2008
```

   In the SQL cell, choose **Run** to get new results.

```
count
-----
14261
```

8. If you ask a question that tries to change the connected database, a warning message is returned along with any recommended SQL. Enter the following prompt to generate SQL to inset data into a table:

```
Insert 1 into the venue table.
```

```


INSERT
,
UPDATE
  OR delete data
FROM
  the database AS that could potentially change the data.Please provide a query
 that ONLY selects data
```

```


I was not able to generate the correct SQL code. I generated SQL, but you'll have
 to edit it to work with your database.
```

If you choose **Add to notebook** to add two cells to the open notebook and run the SQL, then the SQL fails.

```
ERROR: syntax error at or near "," Position: 132 [ErrorId:
 1-6546764a-011df2691778846219ce6ec2]
```

This scenario only illustrated some basic ways to interact with the Amazon Q generative SQL. You can experiment even more with this generative AI technology to help you start authoring SQL to query your database.

Loading data into a database

You can use query editor v2 to load data into a database in an Amazon Redshift cluster or workgroup. This section covers how to load sample data, data from S3, and data from a local file setup and workflow.

Sample data

The query editor v2 comes with sample data and notebooks available to be loaded into a sample database and corresponding schema.

To load sample data, choose the



icon associated with the sample data you want to load. The query editor v2 then loads the data into a schema in database `sample_data_dev` and creates a folder of saved notebooks.

The following sample datasets are available.

tickit

Most of the examples in the Amazon Redshift documentation use sample data called `tickit`. This data consists of seven tables: two fact tables and five dimensions. When you load this data, the schema `tickit` is updated with sample data. For more information about the `tickit` data, see Sample database in the *Amazon Redshift Database Developer Guide*.

tpch

This data is used for a decision support benchmark. When you load this data, the schema `tpch` is updated with sample data. For more information about the `tpch` data, see TPC-H.

tpcds

This data is used for a decision support benchmark. When you load this data, the schema `tpcds` is updated with sample data. For more information about the `tpcds` data, see TPC-DS.

Loading data from Amazon S3

You can load Amazon S3 data into an existing or new table.

To load data into an existing table

The COPY command is used by query editor v2 to load data from Amazon S3. The COPY command generated and used in the query editor v2 load data wizard supports many of the parameters available to the COPY command syntax to copy from Amazon S3. For information about the COPY command and its options used to load data from Amazon S3, see COPY from Amazon Simple Storage Service in the *Amazon Redshift Database Developer Guide*.

1. Confirm that the table is already created in the database where you want to load data.
2. Confirm that you are connected to the target database in the tree-view panel of query editor v2 before continuing. You can create a connection using the context menu (right-click) to the cluster or workgroup where the data will be loaded.

   Choose  **Load data**.
3. For **Data source**, choose **Load from S3 bucket**.
4. In **S3 URIs**, choose **Browse S3** to look for the Amazon S3 bucket that contains the data to load.
5. If the specified Amazon S3 bucket isn't in the same AWS Region as the target table, then choose the **S3 file location** for the AWS Region where the data is located.
6. Choose **This file is a manifest file** if the Amazon S3 file is actually a manifest containing multiple Amazon S3 bucket URIs.
7. Choose the **File format** for the file to be uploaded. The supported data formats are CSV, JSON, DELIMITER, FIXEDWIDTH, SHAPEFILE, AVRO, PARQUET, and ORC. Depending on the specified

file format, you can choose the respective **File options**. You can also select **Data is encrypted** if the data is encrypted and enter the Amazon Resource Name (ARN) of the KMS key used to encrypt the data.

If you choose CSV or DELIMITER, you can also choose the **Delimiter character** and whether to **Ignore header rows** if the specified number of rows are actually column names and not data to load.

8. Choose a compression method to compress your file. The default is no compression.
9. (Optional) The **Advanced settings** support various **Data conversion parameters** and **Load operations**. Enter this information as needed for your file.

   For more information about data conversion and data load parameters, see Data conversion parameters and Data load operations in the *Amazon Redshift Database Developer Guide*.
10. Choose **Next**.
11. Choose **Load existing table**.
12. Confirm or choose the location of the **Target table** including **Cluster or workgroup**, **Database**, **Schema**, and **Table** name where the data is loaded.
13. Choose an **IAM role** that has the required permissions to load data from Amazon S3.
14. (Optional) Choose column names to enter them in **Column mapping** to map columns in the order of the input data file.
15. Choose **Load data** to start the data load.

    When the load completes, the query editor displays with the generated COPY command that was used to load your data. The **Result** of the COPY is shown. If successful, you can now use SQL to select data from the loaded table. When there is an error, query the system view STL_LOAD_ERRORS to get more details. For information about COPY command errors, see STL_LOAD_ERRORS in the *Amazon Redshift Database Developer Guide*.

When you load data into a new table, query editor v2 first creates the table in the database, then loads the data as separate actions in the same workflow.

To load data into a new table

The COPY command is used by query editor v2 to load data from Amazon S3. The COPY command generated and used in the query editor v2 load data wizard supports many of the parameters available to the COPY command syntax to copy from Amazon S3. For information about the COPY

command and its options used to load data from Amazon S3, see COPY from Amazon Simple Storage Service in the *Amazon Redshift Database Developer Guide*.

1. Confirm that you are connected to the target database in the tree-view panel of query editor v2 before continuing. You can create a connection using the context menu (right-click) to the cluster or workgroup where the data will be loaded.

   Choose  **Load data**.

2. For **Data source**, choose **Load from S3 bucket**.
3. In **S3 URIs**, choose **Browse S3** to look for the Amazon S3 bucket that contains the data to load.
4. If the specified Amazon S3 bucket isn't in the same AWS Region as the target table, then choose the **S3 file location** for the AWS Region where the data is located.
5. Choose **This file is a manifest file** if the Amazon S3 file is actually a manifest containing multiple Amazon S3 bucket URIs.
6. Choose the **File format** for the file to be uploaded. The supported data formats are CSV, JSON, DELIMITER, FIXEDWIDTH, SHAPEFILE, AVRO, PARQUET, and ORC. Depending on the specified file format, you can choose the respective **File options**. You can also select **Data is encrypted** if the data is encrypted and enter the Amazon Resource Name (ARN) of the KMS key used to encrypt the data.

   If you choose CSV or DELIMITER, you can also choose the **Delimiter character** and whether to **Ignore header rows** if the specified number of rows are actually column names and not data to load.
7. Choose a compression method to compress your file. The default is no compression.
8. (Optional) The **Advanced settings** support various **Data conversion parameters** and **Load operations**. Enter this information as needed for your file.

   For more information about data conversion and data load parameters, see Data conversion parameters and Data load operations in the *Amazon Redshift Database Developer Guide*.
9. Choose **Next**.
10. Choose **Load new table**.

The table columns are inferred from the input data. You can modify the definition of the table schema by adding columns and table details. To revert to the query editor v2 inferred table schema, choose **Restore to defaults**.

11. Confirm or choose the location of the **Target table** including **Cluster or workgroup**, **Database**, and **Schema** where the data is loaded. Enter a **Table** name to be created.
12. Choose an **IAM role** that has the required permissions to load data from Amazon S3.
13. Choose **Create table** to create the table using the definition shown.

    A review summary of the table definition is displayed. The table is created in the database. To later delete the table, run a DROP TABLE SQL command. For more information, see DROP TABLE in the *Amazon Redshift Database Developer Guide*.
14. Choose **Load data** to start the data load.

    When the load completes, the query editor displays with the generated COPY command that was used to load your data. The **Result** of the COPY is shown. If successful, you can now use SQL to select data from the loaded table. When there is an error, query the system view STL_LOAD_ERRORS to get more details. For information about COPY command errors, see STL_LOAD_ERRORS in the *Amazon Redshift Database Developer Guide*.

Loading data from a local file setup and workflow

You can load data from a local file into an existing or new table.

Administrator setup to load data from a local file

Your query editor v2 administrator must specify the common Amazon S3 bucket in the **Account settings** window. The account users must be configured with the proper permissions.

- Required IAM permissions – the users of load from local file must have `s3:ListBucket`, `s3:GetBucketLocation`, `s3:putObject`, `s3:getObject`, and `s3:deleteObject` permissions. The *`optional-prefix`* can be specified to limit query editor v2 related use of this bucket to objects with this prefix. You might use this option when using this same Amazon S3 bucket for uses other than query editor v2. For more information about buckets and prefixes, see Managing user access to specific folders in the *Amazon Simple Storage Service User Guide*. To make sure that cross user data access is not allowed, we recommend that the query editor v2 administrator use an Amazon S3 bucket policy to restrict object access based on `aws:userid`.

The following example allows Amazon S3 permissions to a *<staging-bucket-name>* with read/write access only to Amazon S3 objects with the `aws:userid` as a prefix.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::<staging-bucket-name>"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::<staging-bucket-name>[/<optional-prefix>]/
${aws:userid}/*"
            ]
        }
    ]
}
```

- Data separation – we recommend that users not have access to each other's data (even briefly). Load from a local file uses the staging Amazon S3 bucket set up by the query editor v2 administrator. Configure the bucket policy for the staging bucket to provide data separation between users. The following example shows a bucket policy that separates data between users of the *<staging-bucket-name>*.

  JSON

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

```
    "Statement": [
        {"Sid": "userIdPolicy",
            "Effect": "Deny",
            "Principal": "*",
            "Action": ["s3:PutObject",
                       "s3:GetObject",
                       "s3:DeleteObject"],
            "NotResource": [
                "arn:aws:s3:::<staging-bucket-name>[/<optional-prefix>]/
${aws:userid}/*"
            ]
         }
    ]
}
```

Loading data from a local file

To load local file data into an existing table

Your query editor v2 administrator must specify the common Amazon S3 bucket in the **Account settings** window. query editor v2 automatically uploads the local file to a common Amazon S3 bucket used by your account, and then uses the COPY command to load data. The COPY command generated and run by the query editor v2 load local file window supports many of the parameters available to the COPY command syntax to copy from Amazon S3. For information about the COPY command and its options used to load data from Amazon S3, see COPY from Amazon S3 in the *Amazon Redshift Database Developer Guide*.

1. Confirm that the table is already created in the database where you want to load data.
2. Confirm that you are connected to the target database in the tree-view panel of query editor v2. You can create a connection using the context menu (right-click) to the cluster or workgroup where the data will be loaded.
3. Choose  **Load data**.
4. For **Data source**, choose **Load from local file**.
5. Choose **Browse** to find the file that contains the data to **Load file**. By default, files with extension `.csv`, `.avro` `.parquet`, and `.orc` are shown, but you can choose other file types. The maximum file size is 100 MB.

6. Choose the **File format** for the file to be uploaded. The supported data formats are CSV, JSON, DELIMITER, FIXEDWIDTH, SHAPEFILE, AVRO, PARQUET, and ORC. Depending on the specified file format, you can choose the respective **File options**. You can also select **Data is encrypted** if the data is encrypted and enter the Amazon Resource Name (ARN) of the KMS key used to encrypt the data.

   If you choose CSV or DELIMITER, you can also choose the **Delimiter character** and whether to **Ignore header rows** if the specified number of rows are actually column names and not data to load.

7. (Optional) The **Advanced settings** support various **Data conversion parameters** and **Load operations**. Enter this information as needed for your file.

   For more information about data conversion and data load parameters, see Data conversion parameters and Data load operations in the *Amazon Redshift Database Developer Guide*.

8. Choose **Next**.

9. Choose **Load existing table**.

10. Confirm or choose the location of the **Target table** including **Cluster or workgroup**, **Database**, **Schema**, and **Table** name where the data is loaded.

11. (Optional) You can choose column names to enter in **Column mapping** to map columns in the order of the input data file.

12. Choose **Load data** to start the data load.

    When the load completes, a message is displayed whether the load was successful or not. If successful, you can now use SQL to select data from the loaded table. When there is an error, query the system view STL_LOAD_ERRORS to get more details. For information about COPY command errors, see STL_LOAD_ERRORS in the *Amazon Redshift Database Developer Guide*.

    The COPY command template that was used to load data appears in your **Query history**. This COPY command template shows some of the parameters used, but it can't be run directly in an editor tab. For more information about query history, see Viewing query and tab history.

When you load data into a new table, query editor v2 first creates the table in the database, then loads the data as separate actions in the same workflow.

To load local file data into a new table

Your query editor v2 administrator must specify the common Amazon S3 bucket in the **Account settings** window. The local file is automatically uploaded to a common Amazon S3 bucket used by your account, and then the COPY command is used by query editor v2 to load data. The COPY command generated and run by the query editor v2 load local file window supports many of the parameters available to the COPY command syntax to copy from Amazon S3. For information about the COPY command and its options used to load data from Amazon S3, see COPY from Amazon S3 in the *Amazon Redshift Database Developer Guide*.

1. Confirm that you are connected to the target database in the tree-view panel of query editor v2. You can create a connection using the context menu (right-click) to the cluster or workgroup where the data will be loaded.
2. Choose  **Load data**.
3. For **Data source**, choose **Load from local file**.
4. Choose **Browse** to find the file that contains the data to **Load file**. By default, files with extension `.csv`, `.avro` `.parquet`, and `.orc` are shown, but you can choose other file types. The maximum file size is 100 MB.
5. Choose the **File format** for the file to be uploaded. The supported data formats are CSV, JSON, DELIMITER, FIXEDWIDTH, SHAPEFILE, AVRO, PARQUET, and ORC. Depending on the specified file format, you can choose the respective **File options**. You can also select **Data is encrypted** if the data is encrypted and enter the Amazon Resource Name (ARN) of the KMS key used to encrypt the data.

   If you choose CSV or DELIMITER, you can also choose the **Delimiter character** and whether to **Ignore header rows** if the specified number of rows are actually column names and not data to load.
6. (Optional) The **Advanced settings** support various **Data conversion parameters** and **Load operations**. Enter this information as needed for your file.

   For more information about data conversion and data load parameters, see Data conversion parameters and Data load operations in the *Amazon Redshift Database Developer Guide*.
7. Choose **Next**.
8. Choose **Load new table**.

9. Confirm or choose the location of the **Target table** including **Cluster or workgroup**, **Database**, and **Schema** where the data is loaded. Enter a **Table** name to be created.
10. Choose **Create table** to create the table using the definition shown.

    A review summary of the table definition is displayed. The table is created in the database. To later delete the table, run a DROP TABLE SQL command. For more information, see DROP TABLE in the *Amazon Redshift Database Developer Guide*.
11. Choose **Load data** to start the data load.

    When the load completes, a message displays indicating whether the load was successful or not. If successful, you can now use SQL to select data from the loaded table. When there is an error, query the system view STL_LOAD_ERRORS to get more details. For information about COPY command errors, see STL_LOAD_ERRORS in the *Amazon Redshift Database Developer Guide*.

    The COPY command template that was used to load data appears in your **Query history**. This COPY command template shows some of the parameters used, but it can't be run directly in an editor tab. For more information about query history, see Viewing query and tab history.

Authoring queries with Amazon Redshift

You can enter a query in the editor or select a saved query from the **Queries** list and choose **Run**.

By default, **Limit 100** is set to limit the results to 100 rows. You can turn off this option to return a larger result set. If you turn off this option, you can include the LIMIT option in your SQL statement if you want to avoid very large result sets. For more information, see ORDER BY clause in the *Amazon Redshift Database Developer Guide*.

To display a query plan in the results area, turn on **Explain**. Turn on **Explain graph** for the results to also display a graphical representation of the explain plan.

To save a query to the **Queries** folder, choose **Save**.

For a successful query, a success message appears. If the query returns information, the results display in the **Results** section. If the number of results exceeds the display area, numbers appear at the top of the results area. You can choose the numbers to display successive pages of results.

You can filter and sort **Result** for each column. To enter filter criteria in the result column header, hover over the column to see a menu

( )

where you can enter criteria to filter the column.

If the query contains an error, the query editor v2 displays an error message in the results area. The message provides information on how to correct the query.

You can export or copy the results of your query by using the context (right-click) menu in the results area as follows:

- Choose **Copy to** and either **JSON** or **CSV** to download the rows that are selected to a file.
- Choose **Copy rows** to copy the selected rows to the clipboard.
- Choose **Copy rows with headers** to copy the selected rows with column headers to the clipboard.

You can also choose **Export** in the results area, then choose either **JSON** or **CSV** to download the entire set of row results to a file. The number of rows in the result set might be limited by the **Limit** option or the SQL `limit` clause in the query. The maximum size of the downloaded result set is 5 MB.

You can also use the shortcut Ctrl+C on Windows or Cmd+C on macOS to copy data from the current results page to the clipboard. If no rows are selected, then the cell with focus is copied to the clipboard. If rows are selected, then the selected rows are copied to the clipboard.

To add a new query tab, choose the



icon, then **Editor**, which appears in the row with the query tabs. The query tab is either using an `Isolated session` or not. With an isolated session, the results of a SQL command, such as creating a temporary table in one editor tab, are not visible in another editor tab. When you open an editor tab in query editor v2, the default is an isolated session.

To run a query

1. In the query area, do one of the following:
   - Enter a query.
   - Paste a query that you copied.
   - Choose the **Queries** folder, open the context menu (right-click) a saved query, and choose **Open query**.

2. Confirm that you chose the correct **Cluster** or **Workgroup**, and **Database** value for the SQL you plan to run.

   Initially, you can choose your **Cluster** or **Workgroup** in the tree view. Choose your **Database** in the tree view also.

   You can change the **Cluster** or **Workgroup**, and **Database** within each editor tab with the drop-down control located near the **Isolated session** header of each editor tab.

   For each editor tab, you choose whether to run the SQL in an **Isolated session**. An isolated session has its own connection to a database. Use it to run SQL that is isolated from other query editor sessions. For more information about connections, see Opening query editor v2.
3. Choose **Run**.

   The **Result** area opens and displays the query results.

To display the explain plan for a query

1. Select the query.
2. Turn on **Explain**.

   By default, the **Explain graph** is also on.
3. Choose **Run**.

   The query runs and the explain plan displays in the query **Result** area.

The query editor v2 supports the following features:

- You can author queries with multiple SQL statements in one query tab. The queries are run serially and multiple results tabs open for each query.
- You can author queries with session variables and temporary tables.
- You can author queries with replaceable parameters designated by `${parameter}`. You can author your SQL query with multiple replaceable parameters and use the same parameter in multiple places in your SQL statement.

  When the query runs, a window is presented to enter the value of the parameter. Each time you run the query, the window is presented to enter your parameter values.

For an example, see Example: Sales greater than a specific parameter.

- Queries are versioned automatically. You can choose an earlier version of a query to run.
- You don't need to wait for a query to complete before continuing on with your workflow. Your queries continue to run even if you close the query editor.
- When authoring queries, automatic completion of schema, table, and column names is supported.

The SQL editor supports the following features:

- The beginning and ending brackets used in SQL have matching colors. Vertical lines are shown in the editor to help you match brackets.
- You can collapse and expand sections of your SQL.
- You can search and replace text in your SQL.
- You can use shortcut keys for several common editing tasks.
- SQL errors are highlighted in the editor for convenient location of problem areas.

For a demo of editing features, watch the following video: New and Enhanced Editing Experience in Amazon Redshift query editor v2.

Query examples

Following, you can find descriptions of the various types of queries that you can run.

The data used in many of these queries is from the `tickit` sample schema. For more information about loading the sample `tickit` data, see Loading data into a database. For more information about the `tickit` sample data, see Sample database in the *Amazon Redshift Database Developer Guide*.

When you run these example queries, confirm that you choose the correct database in the editor, such as `sample_data_dev`.

Topics

- Example: Setting session variables
- Example: Top event by total sales
- Example: Sales greater than a specific parameter

- Example: Create a temporary table
- Example: Selecting from a temporary table

Example: Setting session variables

The following command sets the `search_path` server configuration parameter to *public* for the session. For more information, see SET and search_path in the *Amazon Redshift Database Developer Guide*.

```
set search_path to public;
```

Example: Top event by total sales

The following query finds the event with the most sales.

```
select eventname, count(salesid) totalorders, sum(pricepaid) totalsales
from sales, event
where sales.eventid=event.eventid
group by eventname
order by 3;
```

Following is a partial list of the results.

```
eventname           totalorders       totalsales
White Christmas          20              9352
Joshua Radin             38             23469
Beach Boys               58             30383
Linda Ronstadt           56             35043
Rascal Flatts            76             38214
Billy Idol               67             40101
Stephenie Meyer          72             41509
Indigo Girls             57             45399
...
```

Example: Sales greater than a specific parameter

The following query finds sales where the quantity sold is greater than the parameter specified by `${numberoforders}`. When the parameter value is 7, the result is 60 rows. When you run the query, the query editor v2 displays a **Run query form** window to gather the value of parameters in the SQL statement.

```
select salesid, qtysold
from sales
where qtysold > ${numberoforders}
order by 2;
```

Following is a partial list of the results.

```
salesid qtysold
20005 8
21279 8
130232 8
42737 8
74681 8
67103 8
105533 8
91620 8
121552 8
...
```

Example: Create a temporary table

The following statement creates the temporary table *eventsalestemp* by selecting information from the *sales* and *event* tables.

```
create temporary table eventsalestemp as
select eventname, count(salesid) totalorders, sum(pricepaid) totalsales
from sales, event
where sales.eventid=event.eventid
group by eventname;
```

Example: Selecting from a temporary table

The following statement selects events, total orders, and total sales from the temporary table *eventsalestemp*, ordered by total orders.

```
select eventname,  totalorders,  totalsales
from eventsalestemp
order by 2;
```

Following is a partial list of results.

```
eventname          totalorders   totalsales
White Christmas         20          9352
Joshua Radin            38         23469
Martina McBride         50         52932
Linda Ronstadt          56         35043
Indigo Girls            57         45399
Beach Boys              58         30383
...
```

Notebooks in Amazon Redshift

You can use notebooks to organize, annotate, and share multiple SQL queries in a single document. You can add multiple SQL query and Markdown cells to a notebook. Notebooks provide a way to group queries and explanations associated with a data analysis in a single document by using multiple query and Markdown cells. You can add text and format the appearance using Markdown syntax to provide context and additional information for your data analysis tasks. You can share your notebooks with team members.

To use notebooks, you must add permission for notebooks to your IAM principal (an IAM user or IAM role). As a best practice, we recommend attaching permissions policies to an IAM role and then assigning it to users and groups as needed. For more information, see Identity and access management in Amazon Redshift. You can add the permission to one of the query editor v2 managed policies. For more information, see Accessing the query editor v2.

You can run all the cells of a notebook sequentially. The SQL query cell of a notebook has most of the same capabilities as a query editor tab. For more information, see Authoring queries with Amazon Redshift. The following are differences between a query editor tab and a SQL cell in a notebook.

- There isn't a control to run `Explain` on a SQL statement in a notebook.
- You can create only one chart per SQL cell in a notebook.

You can export and import notebooks to files created with query editor v2. The file extension is `.ipynb` and the file size can be a maximum of 5 MB. The SQL and Markdown cells are stored in the file. A cluster or workgroup and database is not stored in the exported notebook. When you open an imported notebook you choose the cluster or workgroup and the database where to run it. After running SQL cells, you can then choose in the results tab whether to display the current page of results as a chart. The result set of a query is not stored in the notebook.

Creating a notebook

You can create a notebook to organize, annotate, and share multiple SQL queries in a single document.

To create a notebook

1. From the navigator menu, choose the Editor icon ().
2. Choose the plus icon (), and then choose **Notebook**.

   By default, a SQL query cell appears in the notebook.
3. In the SQL query cell, do one of the following:
   - Enter a query.
   - Paste a query that you copied.
4. (Optionally) Choose the plus icon (), then choose **Markdown** to add a Markdown cell where you can provide descriptive or explanatory text using standard Markdown syntax.
5. (Optionally) Choose the plus icon (), then choose **SQL** to insert a SQL cell.

You can rename notebooks using the pencil icon ().

From the menu icon (), you can also perform the following operations on a notebook:

- 

  Share with my team – To share the notebook with your team as defined by tags. To share a notebook with your team, make sure that you have the principal tag `sqlworkbench-team` set to the same value as the rest of your team members in your account. For example, an administrator might set the value to `accounting-team` for everyone in the accounting department. For an example, see Permissions required to use the query editor v2.

- 

  Export – To export the notebook to a local file with the `.ipynb` extension.

- 

  Import query – To import a query from a local file into a cell in the notebook. You can import files with `.sql` and `.txt` extensions.

- 

  Save version – To create a version of the notebook. To see versions of a notebook, navigate to your saved notebooks and open **Version history**.

- 

  Duplicate – To create a copy of the notebook and open it in a new notebook tab.

- 

  Shortcuts – To display the shortcuts available when authoring a notebook.

Importing into notebooks

You can import an entire notebook or individual SQL cells into a query editor v2 notebook.

To import an entire notebook from a local file to **My notebooks**, choose

Import, then choose **Import notebook**. Navigate to the `.ipynb` file that contains your notebook. The notebook is imported into the currently open notebook folder. You can then open the notebook in the notebook editor.

To import a query from a local file into a SQL cell in a notebook, choose

Import, then choose **Import query**. On the **Import query** window, follow the directions on the screen to choose file and folders that can be imported as a query into a new notebook or an

existing notebook. The files must have an extension of `.sql` or `.txt`. Each query can be up to 10,000 characters. When you add to an existing notebook, you choose which notebook from all notebooks in your **Saved notebooks** list. The imported queries are added as SQL cells at the end of the notebook. When you choose a new notebook, you choose the name of the notebook and it is created in the currently open saved notebooks folder.

> **Note**
>
> When creating `.sql` files on macOS using the TextEdit application, you might encounter an issue where an additional hidden extension is added to the file. For instance, a file named `Test.sql` created in TextEdit might end up being saved as `Test.sql.rtf`. The query editor v2 does not support files with the `.rtf` extension. However, if you create a `.sql` file using TextEdit, and save it as a plain text file, the file has an additional hidden `.txt` extension. For example, a file named `Text.sql` might be saved as `Text.sql.txt`. Unlike the `.rtf` extension, query editor v2 does support files with the `.txt` extension, so `Text.sql.txt` is supported when importing queries to notebooks.

For a demo of notebooks, watch the following video: Amazon Redshift SQL Notebooks in query editor v2.

Querying the AWS Glue Data Catalog

You can use query editor v2 to query data cataloged in your AWS Glue Data Catalog by using specific SQL commands and granting the permissions outlined in this section. By default, the AWS Glue Data Catalog is listed as a query editor v2 database named `awsdatacatalog`. Querying the AWS Glue Data Catalog is not available in all Amazon Redshift AWS Regions. Use the SHOW command to determine if this capability is available. For more information about AWS Glue, see What is AWS Glue? in the *AWS Glue Developer Guide*.

> **Note**
>
> Querying the AWS Glue Data Catalog is only supported in Amazon Redshift RG or RA3 node type clusters and Amazon Redshift Serverless.

You can configure your data warehouse and view the AWS Glue database objects cataloged using the following SQL commands:

- SHOW – to display whether `awsdatacatalog` is mounted for the currently connected data warehouse. For example, to show the `data_catalog_auto_mount` parameter value, run:

  ```
  SHOW data_catalog_auto_mount;
  ```

  For more information, see SHOW in the *Amazon Redshift Database Developer Guide*.

- ALTER SYSTEM – to change the system-level configuration of `data_catalog_auto_mount`. For example, to change the `data_catalog_auto_mount` parameter value to on, run:

  ```
  ALTER SYSTEM SET data_catalog_auto_mount = on;
  ```

  The change takes effect when a provisioned cluster is rebooted or a serverless workgroup is automatically paused and resumed. For more information, see ALTER SYSTEM in the *Amazon Redshift Database Developer Guide*.

- SHOW SCHEMAS – shows a list of schemas. The schemas in the database named `awsdatacatalog` represent the AWS Glue databases cataloged in the AWS Glue Data Catalog. For example, to show these schemas, run:

  ```
  SHOW SCHEMAS FROM DATABASE awsdatacatalog;
  ```

  For more information, see SHOW SCHEMAS in the *Amazon Redshift Database Developer Guide*.

- SHOW TABLES – shows a list of tables in a schema. For example, to show the tables in the AWS Glue Data Catalog database named `awsdatacatalog` that are in schema `myglue` run:

  ```
  SHOW TABLES FROM SCHEMA awsdatacatalog.myschema;
  ```

  For more information, see SHOW TABLES in the *Amazon Redshift Database Developer Guide*.

- SHOW COLUMNS – shows a list of columns in a table. For example, to show the columns in the AWS Glue Data Catalog database named `awsdatacatalog` that are in schema `myglue` and table `mytable` run:

  ```
  SHOW COLUMNS FROM TABLE awsdatacatalog.myglue.mytable;
  ```

  For more information, see SHOW COLUMNS in the *Amazon Redshift Database Developer Guide*.

To grant your IAM user or role permission to query the AWS Glue Data Catalog,

1. In the tree-view pane, connect to your initial database in your provisioned cluster or serverless workgroup using the **Database user name and password** authentication method. For example, connect to the `dev` database using the admin user and password you used when you created the cluster or workgroup.
2. In an editor tab, run the following SQL statement to grant an IAM user access to the AWS Glue Data Catalog.

   ```
   GRANT USAGE ON DATABASE awsdatacatalog to "IAM:myIAMUser"
   ```

   Where *`IAM:myIAMUser`* is an IAM user that you want to grant usage privilege to the AWS Glue Data Catalog. Alternatively, you can grant usage privilege to *`IAMR:myIAMRole`* for an IAM role.
3. In the tree-view pane, edit or delete the connection to the cluster or workgroup you previously created. Connect to either your cluster or workgroup in one of the following ways:
   - To access the `awsdatacatalog` database from a cluster, you must use the authentication method **Temporary credentials using your IAM identity**. For more information about this authentication method, see Connecting to an Amazon Redshift database. Your query editor v2 administrator might need to configure the **Account settings** for the account to display this authentication method on the connection window.
   - To access the `awsdatacatalog` database from a workgroup, you must use the authentication method **Federated user**. For more information about this authentication method, see Connecting to an Amazon Redshift database.
4. With the granted privilege, you can use your IAM identity to run SQL against your AWS Glue Data Catalog.

After connecting, you can use query editor v2 to query data cataloged in AWS Glue Data Catalog. On the query editor v2 tree-view pane, choose the cluster or workgroup and `awsdatacatalog` database. In the editor or notebook pane, confirm the correct cluster or workgroup is chosen. The database chosen should be the initial Amazon Redshift database such as `dev`. For information about authoring queries, see Authoring queries with Amazon Redshift and Notebooks in Amazon Redshift. The database named `awsdatacatalog` is reserved to reference the external Data Catalog database in your account. Queries against the `awsdatacatalog` database can only be read-only. Use three-part notation to reference the table in your SELECT statement. Where the first

part is the database name, the second part is the AWS Glue database name, and the third part is the AWS Glue table name.

```
SELECT * FROM awsdatacatalog.<aws-glue-db-name>.<aws-glue-table-name>;
```

You can perform various scenarios that read the AWS Glue Data Catalog data and populate Amazon Redshift tables.

The following example SQL joins two tables that are defined in AWS Glue.

```
SELECT pn.emp_id, alias, role, project_name
FROM "awsdatacatalog"."empl_db"."project_name_table" pn,
"awsdatacatalog"."empl_db"."project_alias_table" pa
WHERE pn.emp_id = pa.emp_id;
```

The following example SQL creates an Amazon Redshift table and populates it with data from a join of two AWS Glue tables.

```
CREATE TABLE dev.public.glue AS
SELECT pn.emp_id, alias, role, project_name
FROM "awsdatacatalog"."empl_db"."project_name_table" pn,
"awsdatacatalog"."empl_db"."project_alias_table" pa
WHERE pn.emp_id = pa.emp_id;
```

Querying Amazon S3 tables (preview)

You can use query editor v2 to query data held in Amazon S3table catalogs mounted to the AWS Glue Data Catalog. Amazon S3 table catalogs are mounted to the AWS Glue Data Catalog on creation, and automatically appear as external databases on all provisioned clusters and serverless workgroups in the same AWS Region under the same account. For more information on accessing Amazon S3 tables using Amazon Redshift, see Accessing Amazon S3 tables with Amazon Redshift in the Amazon Simple Storage Service User Guide.

Querying a data lake

You can query data in an Amazon S3 data lake by following the set of tasks in this tutorial. First, you create an external schema to reference the external database in the AWS Glue Data Catalog. Then, you can query data in the Amazon S3 data lake.

Demo: Query a data lake

For a demo of how to query a data lake, watch the following video. Query your data lake from Amazon Redshift query editor v2.

Prerequisites

Before you work with your data lake in query editor v2, confirm the following was set up in your Amazon Redshift environment:

- Crawl your Amazon S3 data using AWS Glue and enable your Data Catalog for AWS Lake Formation.
- Create an IAM role for Amazon Redshift using the AWS Glue enabled Data Catalog for AWS Lake Formation. For details on this procedure, see To create an IAM role for Amazon Redshift using an AWS Glue Data Catalog enabled for AWS Lake Formation. For more information about using Redshift Spectrum and Lake Formation, see Using Redshift Spectrum with AWS Lake Formation.
- Grant SELECT permissions on the table to query in the Lake Formation database. For details on this procedure, see To grant SELECT permissions on the table to query in the Lake Formation database.

  You can verify in the Lake Formation console (https://console.aws.amazon.com/lakeformation/), **Permissions** section, **Data lake permissions** page, that the IAM role, AWS Glue database, and tables have the proper permissions.
- Confirm your connected user has permission to create schemas in the Amazon Redshift database and access data in your data lake. When you connect to a database in query editor v2, you choose an authentication method that includes credentials, which can be a database user or IAM user. The connected user must have the proper permissions and database privileges, such as a `superuser`. The Amazon Redshift `admin` user who created the cluster or workgroup has `superuser` privileges and can create schemas and manage the Redshift database. For more information about connecting to a database with query editor v2, see Connecting to an Amazon Redshift database.

Creating an external schema

To query data in an Amazon S3 data lake, first create an external schema. The external schema references the external database in the AWS Glue Data Catalog.

1. In the **Editor** view of query editor v2, choose

   Create

   and then choose **Schema**.
2. Enter a **Schema** name.
3. For **Schema type**, choose **External**.
4. Within **Data Catalog** details, the **Region** defaults to the AWS Region where your Redshift database is located.
5. Choose the **AWS Glue database** that the external schema will map to and that contains references to the AWS Glue tables.
6. Choose an **IAM role** for Amazon Redshift that has the required permissions to query data on Amazon S3.
7. Optionally, choose an **IAM role** that has permission to the Data Catalog.
8. Choose **Create schema**.

   The schema appears under your database in the tree-view panel.

When creating the schema, if you receive a permission denied error for your database, check if the connected user has the database privilege to create a schema.

Querying data in your Amazon S3 data lake

You use the schema that you created in the previous procedure.

1. In the tree-view panel, choose the schema.
2. To view a table definition, choose a table. The table columns and data types display.
3. To query a table, choose the table and in the context menu (right-click), choose **Select table** to generate a query.
4. Run the query in the **Editor**.

   The following example SQL was generated by query editor v2 to query all the rows in AWS Glue table named `flightscsv`. The columns and rows shown in the output are truncated for simplicity.

```
SELECT * FROM "dev"."mydatalake_schema"."flightscsv";
```

```
year    quarter   month   dom  day_of_week   fl_date    unique_carrier  airline_id
  carrier   tail_num   fl_num
2016    4         10      19   3             10/19/16   OO              20304
  OO         N753SK    3086
2016    4         10      19   3             10/19/16   OO              20304
  OO         N753SK    3086
2016    4         10      19   3             10/19/16   OO              20304
  OO         N778SK    3087
2016 4          10      19   3             10/19/16   OO              20304
 OO         N778SK    3087
...
```

Datashares

You can create a datashare so that users on another cluster can query the data. The cluster containing the data that you want to share is called the *producer* cluster. You create a datashare on the producer cluster for the database objects that you want to share. You can share schemas, tables, views, and SQL user-defined functions (UDFs). The cluster that you want to share the data to is called the *consumer* cluster. On the consumer cluster, you create a database from the datashare. Then, users on the consumer cluster can query the data. For more information, see Getting started with data sharing in the *Amazon Redshift Database Developer Guide*.

Creating datashares

You create a datashare on the cluster that you want to use as the producer cluster. To learn more about datashare considerations, see Data sharing considerations in Amazon Redshift in the *Amazon Redshift Database Developer Guide*.

1. Choose the database on the producer cluster that you want to use.
2. Create the datashare. For example:

```
create datashare mysource;
```

3. Set permissions on the datashare. For example:

```
grant alter, share on datashare mysource to admin;
```

4. Set permissions on the database objects that you want to share. For example:

```
alter datashare mysource add schema public;
```

```
alter datashare mysource add table public.event;
```

5. Set permissions on the consumer cluster namespace to access the datashare. For example:

```
grant usage on datashare mysource to namespace '2b12345-1234-5678-9012-
bb1234567890';
```

Showing datashares

You can show the datashares that you've created on the producer cluster.

1. Choose the producer cluster.
2. Show the datashares. For example:

```
show datashares;
```

```
share_name share_owner source_database  consumer_database share_type createdate
 is_publicaccessible share_acl producer_account producer_namespace
test_datashare 100  db_producer  NULL   OUTBOUND 2/15/2022  FALSE  admin
 123456789012  p1234567-8765-4321-p10987654321
```

Creating the consumer database

On the consumer cluster, you create a database from the datashare. These steps describe how to share data between two clusters in the same account. For information on sharing data across AWS accounts, see Sharing data across AWS accounts in the *Amazon Redshift Database Developer Guide*.

You can use SQL commands or the query editor v2 tree-view panel to create the database.

To use SQL

1. Create a database from the datashare for your account and the namespace of the producer cluster. For example:

```
create database share_db from datashare mysource of account '123456789012'
 namespace 'p1234567-8765-4321-p10987654321';
```

2. Set permissions so that users can access the database and the schema. For example:

```
grant usage on database share_db to usernames;
```

```
grant usage on schema public to usernames;
```

To use the query editor v2 tree-view panel

1. Choose

Create
and then choose **Database**.
2. Enter a **Database name**.
3. (Optional) Select **Users and groups**, and choose a **Database user**.
4. Choose **Create using a datashare**.
5. Choose the datashare.
6. Choose **Create database**.

   The new

datash
   database displays in the query editor v2 tree-view panel.
7. Set permissions so that users can access the database and the schema. For example:

```
grant usage on database share_db to usernames;
```

```
grant usage on schema public to usernames;
```

Querying datashare objects

On the consumer cluster, you can query datashare objects using fully qualified object names expressed with the three-part notation: database, schema, and name of the object.

1. In the query editor v2 tree-view panel, choose the schema.
2. To view a table definition, choose a table.

   The table columns and data types display.
3. To query a table, choose the table and use the context menu (right-click) to choose **Select table**.
4. Query tables using SELECT commands. For example:

```
select top 10 * from test_db.public.event;
```

Scheduled queries with query editor v2

With Amazon Redshift query editor v2, you can automate SQL queries to run on a schedule. Scheduled queries are SQL statements that run automatically at specified times or intervals, letting you efficiently manage recurring data operations and analytics tasks. You might want to schedule queries if you're seeking to streamline batch processing, generate regular reports, or maintain data pipelines within their Amazon Redshift environment.

Scheduled queries facilitate automating extract, transform, and load (ETL) workflows, refreshing dashboards with up-to-date insights, and operationalizing various data management routines. The following pages detail the process of creating, configuring, and managing scheduled queries to optimize your Amazon Redshift workloads.

Creating a query schedule with query editor v2

You can create a schedule to run a SQL statement with Amazon Redshift query editor v2. You create a schedule to run your SQL statement at the time intervals that match your business needs. When it's time for the scheduled query to run, the query is started by Amazon EventBridge and uses the Amazon Redshift Data API.

To create a schedule to run a SQL statement

1. On the **Editor**

   view, choose



Schedule to create a schedule to run a SQL statement.

2. When you define the schedule, you provide the following information.

   - The IAM role that assumes the required permissions to run the query. This IAM role is also attached to your cluster or workgroup.
   - The authentication values for either AWS Secrets Manager or temporary credentials to authorize access your cluster or workgroup. These authentication methods are supported by the Data API. For more information, see Authenticating a scheduled query.
   - The cluster or workgroup where your database resides.
   - The name of the database that contains the data to be queried.
   - The name of the scheduled query and its description. The query editor v2 prefixes the scheduled query name you provide with "QS2-". The query editor v1 prefixes its scheduled query names with "QS-".
   - The SQL statement to be run on the schedule.
   - The schedule frequency and repeat options or a cron formatted value that defines the schedule. For more information, see Cron Expressions in the *Amazon CloudWatch Events User Guide*.
   - Optionally, you can enable standard Amazon SNS notifications to monitor the scheduled query. You might need to confirm the email address you provide to the Amazon SNS notification. Check your email for a link to confirm the email address for the Amazon SNS notification. For more information, see Email notifications in the *Amazon Simple Notification Service Developer Guide*. If your query is being run but you don't see messages published in your SNS topic, see My rule runs, but I don't see any messages published into my Amazon SNS topic in the *Amazon EventBridge User Guide*.

3. Choose **Schedule query** to save and activate the schedule and add the schedule to the list of queries in the **Scheduled queries** view.

The **Scheduled queries**



view lists all the scheduled queries for your clusters and workgroups. With this view, you can display schedule query details, activate or deactivate the schedule, edit the schedule, and delete

the scheduled query. When you view query details, you can also view the history of running the query with the schedule.

> **Note**
>
> A schedule query run is only available in the **Schedule history** list for 24 hours. Queries that run on a schedule don't appear in the **Query history** view of query editor v2.

Demo of scheduling a query

For a demo of scheduling a query, watch the following video.Video demo of scheduling a query.

Setting up permissions to schedule a query

To schedule queries, the AWS Identity and Access Management (IAM) user defining the schedule and the IAM role associated with the schedule must be configured with the IAM permissions to use Amazon EventBridge and Amazon Redshift Data API. To receive emails from scheduled queries, the Amazon SNS notification you optionally specify must be configured also.

The following describes the tasks to use AWS managed policies to provide permission, but depending on your environment, you might want to scope down the permissions allowed.

For the IAM user logged into query editor v2, edit the IAM user using the IAM console (https://console.aws.amazon.com/iam/).

- In addition to permissions to run Amazon Redshift and query editor v2 operations, attach the `AmazonEventBridgeFullAccess` and `AmazonRedshiftDataFullAccess` AWS managed policies to an IAM user.
- Alternatively, assign the permissions to a role and assign the role to the user.

  Attach a policy that allows the `sts:AssumeRole` permission to the resource ARN of the IAM role you specify when you define the scheduled query. For more information about assuming roles, see Granting a user permissions to switch roles in the *IAM User Guide*.

  The following example shows a permission policy that assumes the IAM role `myRedshiftRole` in account `123456789012`. The IAM role `myRedshiftRole`is also the IAM role that is attached to the cluster or workgroup where the scheduled query runs.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AssumeIAMRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": [
                "arn:aws:iam::123456789012:role/myRedshiftRole"
            ]
        }
    ]
}
```

Update the trust policy of the IAM role used to schedule the query to allow the IAM user to assume it.

```
{
            "Sid": "AssumeRole",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/myIAMusername"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

For the IAM role that you specify to allow the scheduled query to run, edit the IAM role using the IAM console (https://console.aws.amazon.com/iam/).

- Attach the `AmazonRedshiftDataFullAccess` and `AmazonEventBridgeFullAccess` AWS managed policies to the IAM role. The `AmazonRedshiftDataFullAccess` managed policy only allows `redshift-serverless:GetCredentials` permission for Redshift Serverless workgroups that are tagged with the key `RedshiftDataFullAccess`.

Authenticating a scheduled query

When you schedule a query, you use one of the following authentication methods when the SQL runs. Each method requires a different combination of input on the query editor v2. These authentication methods are supported by the Data API which is used to run your SQL statements.

The database user or role that is used to run the query must have the necessary database privileges. For example, to grant `IAMR:MyRedshiftQEv2Scheduler` privileges to table `mytable`, run the following SQL command.

```
GRANT all ON TABLE mytable TO "IAMR:MyRedshiftQEv2Scheduler";
```

To view the list of database users in your cluster or workgroup, query the system view `PG_USER_INFO`.

Note

Any Redshift Serverless workgroup for which you schedule queries much be tagged with the key `RedshiftDataFullAccess`. For more information, see Authorizing access to the Amazon Redshift Data API.
As an alternative to tagging the workgroup, you can add an inline policy to the IAM role (that is specified with the schedule) that allows `redshift-serverless:GetCredentials`. For example:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "UseTemporaryCredentialsForAllServerlessWorkgroups",
            "Effect": "Allow",
            "Action": "redshift-serverless:GetCredentials",
            "Resource": [
                "arn:aws:redshift-serverless:*:*:workgroup/*"
            ]
        }
    ]
}
```

AWS Secrets Manager

With this method, provide a secret value for **secret-arn** that is stored in AWS Secrets Manager. This secret contains credentials to connect to your database. You might have created a secret with the proper credentials when you created your cluster or workgroup. The secret must be tagged with the key `RedshiftDataFullAccess`. If the tag key is not already present, use the AWS Secrets Manager console to add it. For information about creating a secret, see Creating a secret for database connection credentials.

For more information about the minimum permissions, see Creating and Managing Secrets with AWS Secrets Manager in the *AWS Secrets Manager User Guide*.

Temporary credentials

With this method, provide your **Database name** and **Database user** values when connecting to a database in a cluster. You only need to provide your **Database name** when connecting to a database in a workgroup.

When connecting to a cluster, the `AmazonRedshiftDataFullAccess` policy allows the database user named `redshift_data_api_user` permission for `redshift:GetClusterCredentials`. If you want to use a different database user to run the SQL statement, then add a policy to the IAM role attached to your cluster to allow `redshift:GetClusterCredentials`. The following example policy allows database users `awsuser` and `myuser`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "UseTemporaryCredentialsForAllDbUsers",
            "Effect": "Allow",
            "Action": "redshift:GetClusterCredentials",
            "Resource": [
                "arn:aws:redshift:*:*:dbuser:*/awsuser",
                "arn:aws:redshift:*:*:dbuser:*/myuser"
            ]
        }
    ]
}
```

Setting up permissions to view schedule query history

To allow users to view schedule query history, edit the IAM role (that is specified with the schedule) **Trust relationships** to add permissions.

The following is an example of a trust policy in an IAM role that allows the IAM user *myIAMusername* to view schedule query history. Instead of allowing an IAM user `sts:AssumeRole` permission you can choose to allow an IAM role this permission.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "redshift.amazonaws.com",
                    "redshift-serverless.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Sid": "AssumeRole",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/myIAMusername"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Monitoring the scheduled query

For the Amazon SNS topic that you specify to send email notifications, create the Amazon SNS topic using the query editor v2 by navigating to the **SNS notifications** section, **Turn on** monitoring, and create the topic with **Create SNS topic**. The query editor v2 creates the Amazon SNS topic and adds a service principal to the access policy for Amazon EventBridge. The following is an example **Access policy** that is created in the Amazon SNS topic. In the example, the AWS Region *us-west-2*, AWS account *123456789012*, and Amazon SNS topic *select-version-pdx-testunload* are used.

JSON

```
{
  "Version":"2012-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "Allow_Publish_Events",
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-west-2:123456789012:select-version-pdx-testunload"
    }
  ]
}
```

When the scheduled query runs, Amazon SNS sends AWS notification emails. The following example shows an email sent to *myemail@example.com* for scheduled query *QS2-may25a* that ran on AWS Region *eu-north-1* in AWS account *123456789012* using Amazon SNS notification topic *may25a-SNS*.

```
{"version":"0","id":"8e4323ec-5258-7138-181b-91290e30ff9b","detail-type":"Scheduled
 Event","source":"aws.events","account":"123456789012","time":"2023-05-25T15:22:00Z",
                   "region":"eu-north-1","resources":["arn:aws:events:eu-
north-1:123456789012:rule/QS2-may25a"],"detail":{}}
```

```
--
If you wish to stop receiving notifications from this topic, please click or visit the
 link below to unsubscribe:
https://sns.eu-north-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:eu-
north-1:123456789012:may25a-SNS:0c1a3d05-39c2-4507-
bc3d-47250513d7b0&Endpoint=myemail@example.com

Please do not reply directly to this email. If you have any questions or comments
 regarding this email, please contact us at https://aws.amazon.com/support
```

Troubleshooting set up of scheduling a query

Consider the following if you have issues scheduling a query.

Queries not running

Check if the IAM role used in the schedule has permission to get the temporary cluster credentials. The permission for provisioned clusters is `redshift:GetClusterCredentialsWithIAM`. The permission for Redshift Serverless workgroups is `redshift-serverless:GetCredentials`.

Scheduled history not displaying

The IAM user or IAM role used to log in to the AWS console was not added into the trust policy of the IAM role used to schedule the query.

When using AWS Secrets Manager for the scheduled query to connect, confirm the secret is tagged with the key `RedshiftDataFullAccess`.

If the scheduled query is using an AWS Secrets Manager connection, the IAM role used to schedule the query must have the equivalent of managed policy `SecretsManagerReadWrite` attached to the role.

Query history status is `Failed`

View the SYS_QUERY_HISTORY system view for details about why the query failed. A common issue is that the database user or role that was used to run the query might not have the required privilege to run the SQL. For more information, see Authenticating a scheduled query.

The following SQL queries the SYS_QUERY_HISTORY view to return failed queries.

```
SELECT user_id, query_id, transaction_id, session_id, database_name, query_type,
 status, error_message, query_text
FROM sys_query_history
```

```
WHERE status = 'failed';
```

To find out details for a specific failing scheduled query, see Viewing the results of a scheduled query with AWS CloudShell.

Viewing the results of a scheduled query with AWS CloudShell

You can use AWS CloudShell to find out details about a schedule query. You must have the proper permissions to run the AWS CLI commands shown in the following procedure.

To view the results of a scheduled query

1. On the AWS console, open the AWS CloudShell command prompt. For more information about AWS CloudShell, see What is AWS CloudShell in the *AWS CloudShell User Guide*.
2. Assume the IAM role of the scheduled query. To assume the role, find the IAM role associated with the scheduled query in query editor v2 and use it in the AWS CLI command in AWS CloudShell. For example, for the role `scheduler` enter an AWS STS command to assume the role used by the scheduled query.

   ```
   aws sts assume-role --role-arn "arn:aws:iam::123456789012:role/scheduler" --role-
   session-name "scheduler-test"
   ```

   The credentials returned are similar to the following.

   ```
   "Credentials": {
   "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
   "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
   "SessionToken": "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY...",
   "Expiration": "2023-08-18T18:19:44+00:00"
   },
   "AssumedRoleUser": {
   "AssumedRoleId": "AROA35B2NH6WBTP7ONL4E:scheduler-test",
   "Arn": "arn:aws:sts::123456789012:assumed-role/scheduler/scheduler-test"
   }
   }
   ```

3. Create environmental variables in the AWS CLI using the credentials displayed from assuming the IAM role. You must use these tokens before their expiration time. For example, you enter the following in AWS CloudShell.

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
export AWS_SESSION_TOKEN=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY...
```

4. To view the error of a failed query, run the AWS CLI command to describe a statement. The id of the SQL statement is from the **ID** shown in the **Schedule history** section of a scheduled query in the query editor v2.

```
aws redshift-data describe-statement --id 130d2620-05d2-439c-b7cf-815d9767f513
```

In this example, the scheduled SQL `select * from users limit 100` results in a SQL error that the `users` table does not exist.

```
{
"CreatedAt": "2023-08-18T17:39:15.563000+00:00",
"Duration": -1,
"Error": "ERROR: relation \"users\" does not exist",
"HasResultSet": false,
"Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
"QueryString": "select * from users limit 100\n—RequestID=a1b2c3d4-5678-90ab-cdef-
EXAMPLE22222; TraceID=1-633c5642-4039308d03f3a0ba53dbdf6f",
"RedshiftPid": 1073766651,
"RedshiftQueryId": 0,
"ResultRows": -1,
"ResultSize": -1,
"Status": "FAILED",
"UpdatedAt": "2023-08-18T17:39:16.116000+00:00",
"WorkgroupName": "default"
}
```

Visualizing query results

After you run a query and the results display, you can turn on **Chart** to display a graphic visualization of the current page of results. You can use the following controls to define the content, structure, and appearance of your chart:



Trace

Represents a set of related graphical marks in a chart. You can define multiple traces in a chart.

Type

You can define the trace type to represent data as one of the following:

- Scatter chart for a scatter plot or bubble chart.
- Bar chart to represent categories of data with vertical or horizontal bars.
- Area chart to define filled areas.
- Histogram that uses bars to represent frequency distribution.
- Pie chart for a circular representation of data where each slice represents a percentage of the whole.
- Funnel or Funnel Area chart to represent data through various stages of a process.
- OHLC (open-high-low-close) chart often used for financial data to represent open, high, low, and close values along the x-axis, which usually represents intervals of time.
- Candlestick chart to represent a range of values for a category over a timeline.
- Waterfall chart to represent how an initial value increases or decreases through a series of intermediate values. Values can represent time intervals or categories.
- Line chart to represent changes in value over time.

X axis

You specify a table column that contains values to plot along the X axis. Columns that contain descriptive values usually represent dimensional data. Columns that contain quantitative values usually represent factual data.

Y axis

You specify a table column that contains values to plot along the Y axis. Columns that contain descriptive values usually represent dimensional data. Columns that contain quantitative values usually represent factual data.

Subplots

You can define additional presentations of chart data.

Transforms

You can define transforms to filter trace data. You use a split transform to display multiple traces from a single source trace. You use an aggregate transform to present a trace as an average or minimum. You use a sort transform to sort a trace.

General appearance

You can set defaults for background color, margin color, color scales to design palettes, text style and sizes, title style and size, and mode bar. You can define interactions for drag, click, and hover. You can define meta text. You can define default appearances for traces, axes, legends, and annotations.

To create a chart

1. Run a query and get results.
2. Turn on **Charts**.
3. Choose **Trace** and start to visualize your data.
4. Choose a chart style from one of the following:
   - Scatter
   - Bar
   - Area
   - Histogram
   - Pie
   - Funnel
   - Funnel Area
   - OHLC (open-high-low-close)
   - Candlestick
   - Waterfall
   - Line
5. Choose **Style** to customize the appearance, including colors, axes, legend, and annotations. You can add text, shapes, and images.
6. Choose **Annotations** to add text, shapes, and images.
7. Choose **Refresh** to update the chart display. Choose **Full screen** to expand the chart display.

Example: Create a pie chart to visualize query results

The following example uses the *Sales* table of the sample database. For more information, see Sample database in the *Amazon Redshift Database Developer Guide*.

Following is the query that you run to provide the data for the pie chart.

```
select top 5 eventname, count(salesid) totalorders, sum(pricepaid) totalsales
from sales, event
where sales.eventid=event.eventid group by eventname
order by 3;
```

To create a pie chart for the top event by total sales

1. Run the query.
2. In the query results area, turn on **Chart**.
3. Choose **Trace**.
4. For **Type**, choose **Pie**.
5. For **Values**, choose *totalsales*.
6. For **Labels**, choose *eventname*.
7. Choose **Style** and then **General**.
8. Under **Colorscales**, choose **Categorical** and then **Pastel2**.



Example: Create a combination chart for comparing revenue and sales

Perform the steps in this example to create a chart that combines a bar chart for revenue data and a line graph for sales data. The following example uses the *Sales* table of the tickit sample

database. For more information, see Sample database in the *Amazon Redshift Database Developer Guide*.

Following is the query that you run to provide the data for the chart.

```
select eventname, total_price, total_qty_sold
from  (select eventid, total_price, total_qty_sold, ntile(1000) over(order by
 total_price desc) as percentile
       from (select eventid, sum(pricepaid) total_price, sum(qtysold) total_qty_sold
             from   tickit.sales
             group by eventid)) Q, tickit.event E
       where Q.eventid = E.eventid
       and percentile = 1
order by total_price desc;
```

To create a combination chart for comparing revenue and sales

1. Run the query.
2. In the query results area, turn on **Chart**.
3. Under *trace o*, for **Type**, choose **Bar**.
4. For **X**, choose *eventname*.
5. For **Y**, choose *total_price*.

   The bar chart displays with event names along the X axis.
6. Under **Style**, choose **Traces**.
7. For **Name**, enter *Revenue*.
8. Under **Style**, choose **Axes**.
9. For **Titles**, choose **Y** and enter *Revenue*.

   The label *Revenue* displays on the left Y axis.
10. Under **Structure**, choose **Traces**.
11. Choose

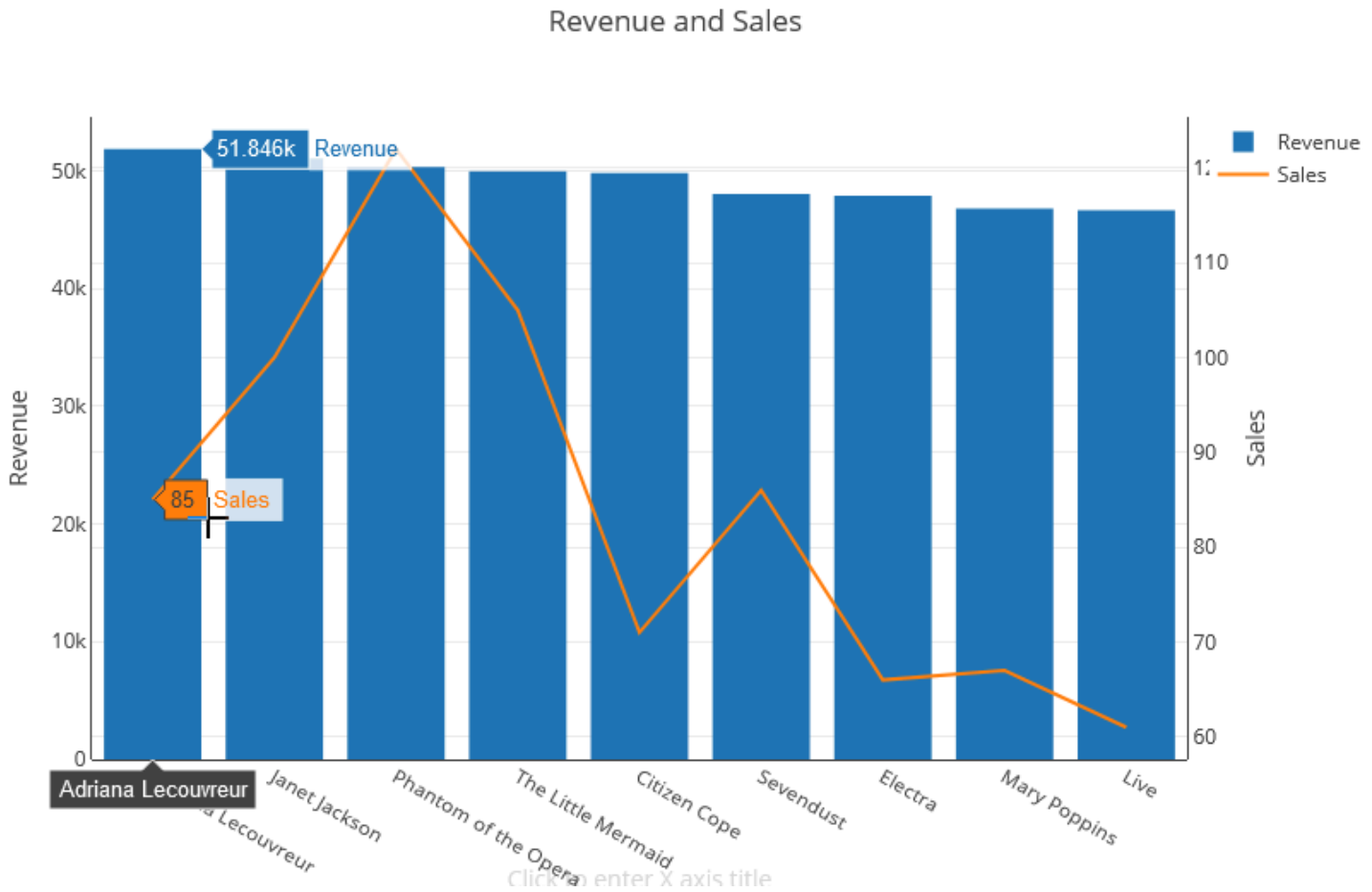
    

    Trace.

    The trace 1 options display.
12. For **Type**, choose **Line**.

13. For **X**, choose *eventname*.
14. For **Y**, choose *total_qty_sold*.
15. Under **Axes To Use**, for **Y Axis** choose
    +
    .

    The **Y Axis** displays *Y2*.
16. Under **Style**, choose **Axes**.
17. Under **Titles**, choose **Y2**.
18. For **Name**, enter *Sales*.
19. Under **Lines**, choose *Y:Sales*.
20. Under **Axis Line**, choose **Show** and for **Position**, choose **Right**.



Demo: Build visualizations using Amazon Redshift query editor v2

For a demo of how to build visualizations, watch the following video. Build visualizations using Amazon Redshift query editor v2.

Collaborating and sharing as a team

You can share queries with your team.

A team is defined for a set of users who collaborate and share query editor v2 resources. An administrator can create a team by adding a tag to an IAM role. For more information, see Permissions required to use the query editor v2.

Saving and browsing for queries

Before you can share your query with your team, save your query. You can view and delete saved queries.

To save a query

1. Prepare your query and choose **Save**.
2. Enter a title for your query.
3. Choose **Save**.

To browse for saved queries

1. Choose **Queries** from the navigation pane.
2. You can view queries that are **My queries**, **Shared by me**, or **Shared to my team**. These queries can appear as individual queries or within folders you created.

Sharing a query

You can share your queries with your team. You can also view the history of saved queries and manage query versions.

To share a query with your team, make sure that you have the principal tag `sqlworkbench-team` set to the same value as the rest of your team members in your account. For example, an administrator might set the value to `accounting-team` for everyone in the accounting department. For an example, see Permissions required to use the query editor v2.

To share a query with a team

1. Choose **Queries** from the navigation pane.
2. Open the context (right-click) menu of the query that you want to share and choose **Share with my team**.
3. Choose the team or teams that you want to share the query with and then choose **Save sharing options**.

Managing query versions

Every time you save a SQL query, the query editor v2 saves it as a new version. You can browse earlier query versions, save a copy of a query, or restore a query.

To manage query versions

1. Choose **Queries** from the navigation pane.
2. Open the context (right-click) menu for the query that you want to work with.
3. Choose **Version history** to open a list of versions of the query.
4. On the **Version history** page, you can do the following:
   - **Revert to selected** – Revert to the selected version and continue your work with this version.
   - **Save selected as** – Create a new query in the editor.

Querying a database using the Amazon Redshift query editor v1

Using the query editor is an easy way to run queries on databases hosted by your Amazon Redshift cluster. After creating your cluster, you can immediately run queries by using the query editor on the Amazon Redshift console.

> **Note**
>
> You can't query data in Amazon Redshift Serverless using this original query editor. Use Amazon Redshift query editor v2 instead.

In February 2021, an updated query editor was deployed and authorization permissions to use the query editor changed. The new query editor uses the Amazon Redshift Data API to run queries. The `AmazonRedshiftQueryEditor` policy, which is an AWS managed AWS Identity and Access Management (IAM) policy, was updated to include the necessary permissions. If you have a custom IAM policy, make sure that you update it. Use `AmazonRedshiftQueryEditor` as a guide. The changes to `AmazonRedshiftQueryEditor` include the following:

- Permission to manage query editor statement results requires the statement owner user.
- Permission to use Secrets Manager to connect to a database has been added.

For more information, see Permissions required to use the Amazon Redshift console query editor.

When you connect to your cluster from the new query editor, you can use one of two authentication methods.

Using the query editor, you can do the following:

- Run single SQL statement queries.
- Download result sets as large as 100 MB to a comma-separated value (CSV) file.
- Save queries for reuse. You can't save queries in the Europe (Paris) Region, the Asia Pacific (Osaka) Region, the Asia Pacific (Hong Kong) Region, or the Middle East (Bahrain) Region.
- View query runtime details for user-defined tables.
- Schedule queries to run at a future time.
- View a history of queries that you created in the query editor.
- Run queries against clusters using enhanced VPC routing.

Query editor considerations

Consider the following about working with queries when you use the query editor:

- The maximum duration of a query is 24 hours.
- The maximum query result size is 100 MB. If a call returns more than 100 MB of response data, the call is terminated.
- The maximum retention time for query results is 24 hours.
- The maximum query statement size is 100 KB.

- The cluster must be in a virtual private cloud (VPC) based on the Amazon VPC service.
- You can't use transactions in the query editor. For more information about transactions, see BEGIN in the *Amazon Redshift Database Developer Guide.*
- You can save a query up to 3,000 characters long.

Connecting to an Amazon Redshift data warehouse using SQL client tools

You can connect to Amazon Redshift data warehouses from SQL client tools over Java Database Connectivity (JDBC), Python, and Open Database Connectivity (ODBC) connections. Amazon Redshift doesn't provide or install any SQL client tools or libraries. To use these tools or libraries to work with data in your data warehouses, install them on your client computer or Amazon EC2 instance. You can use most SQL client tools that support JDBC, Python, or ODBC drivers.

Use the list of sections at the end of this topic to help you walk through the process of configuring your client computer or Amazon EC2 instance to use a JDBC, Python, or ODBC connection. The topics also discuss related security options for the client connection to the server. Additionally, find information about setting up and connecting from SQL client tools, such as Amazon Redshift RSQL. You can try these tools if you don't yet have a business intelligence tool to use. You can also use this section to learn about connecting to your data . Finally, if you encounter issues when attempting to connect to your data warehouse, you can review the troubleshooting information to identify solutions.

Recommendations for connecting with client tools

If you connect to your Redshift cluster using an IP address, it can result in additional downtime when there is an outage or a connection loss and the cluster is brought online in a new Availability Zone (AZ). However, if you still want your application to connect to Redshift using an IP address, use the private IP address attached to the cluster's virtual-private-cloud (VPC) endpoint. You can find this in the cluster details in **Network and security**, under the **Properties** tab.

> **Note**
>
> If your application uses the IP address of the leader node to access the Redshift cluster, the recommended best practice is to change it to use the cluster endpoint URL. For more information, see Configuring connections in Amazon Redshift.

Topics

- Configuring connections in Amazon Redshift
- Best practices for discovering metadata
- Configuring security options for connections
- Connecting from client tools and code
- Using an authentication profile to connect to Amazon Redshift
- Troubleshooting connection issues in Amazon Redshift

Configuring connections in Amazon Redshift

In the following section, learn how to configure JDBC, Python, and ODBC connections to connect to your cluster from SQL client tools. This section describes how to set up JDBC, Python, and ODBC connections. It also describes how to use Secure Sockets Layer (SSL) and server certificates to encrypt communication between the client and server.

JDBC, Python, and ODBC drivers for Amazon Redshift

To work with data in your cluster, you must have JDBC, Python, or ODBC drivers for connectivity from your client computer or instance. Code your applications to use JDBC, Python, or ODBC data access API operations, and use SQL client tools that support either JDBC, Python, or ODBC.

Amazon Redshift offers JDBC, Python, and ODBC drivers for download. These drivers are supported by Support. PostgreSQL drivers are not tested and not supported by the Amazon Redshift team. Use the Amazon Redshift–specific drivers when connecting to an Amazon Redshift cluster. The Amazon Redshift drivers have the following advantages:

- Support for IAM, SSO, and federated authentication.
- Support for new Amazon Redshift data types.
- Support for authentication profiles.
- Improved performance in conjunction with Amazon Redshift enhancements.

For more information about how to download the JDBC and ODBC drivers and configure connections to your cluster, see Configuring a connection for JDBC driver version 2.x for Amazon Redshift, Amazon Redshift Python connector, and Configuring a connection for ODBC driver version 2.x for Amazon Redshift.

For more information about managing IAM identities, including best practices for IAM roles, see Identity and access management in Amazon Redshift.

Finding your cluster connection string

To connect to your cluster with your SQL client tool, you must have the cluster connection string. You can find the cluster connection string in the Amazon Redshift console, on a cluster's details page.

To find the connection string for a cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the cluster name from the list to open its details.
3. The **JDBC URL** and **ODBC URL** connection strings are available, along with additional details, in the **General information** section. Each string is based on the AWS Region where the cluster runs. Click the icon next to the appropriate connection string to copy it.

To connect to a cluster endpoint, you can use the cluster endpoint URL from a DescribeClusters API request. The following is an example of a cluster endpoint URL.

```
mycluster.cmeaswqeuae.us-east-2.redshift.amazonaws.com
```

If you have set up a custom domain name for your cluster, you can also use that to connect to your cluster. For more information about creating a custom domain name, see Setting up a custom domain name.

> **Note**
>
> When you connect, don't use the IP address of a cluster node or the IP address of the VPC endpoint. Always use the Redshift endpoint to avoid an unnecessary outage. The only exception to using the endpoint URL is when you use a custom domain name. For more information, see Using a custom domain name for client connections.

Configuring a connection for JDBC driver version 2.x for Amazon Redshift

You can use a JDBC driver version 2.x connection to connect to your Amazon Redshift cluster from many third-party SQL client tools. The Amazon Redshift JDBC connector provides an open source solution. You can browse the source code, request enhancements, report issues, and provide contributions.

For the latest information about JDBC driver changes, see the change log.

By default, the Amazon Redshift JDBC driver is configured to use TCP keepalives to prevent connections from timing out. You can specify when the driver starts sending keepalive packets or turn off the feature by setting the relevant properties in the connection URL. For more information about the syntax of the connection URL, see Building the connection URL.

Property	Description
`TCPKeepAlive`	To turn off TCP keepalives, set this property to `FALSE`.

Topics

- Download the Amazon Redshift JDBC driver, version 2.x
- Installing the Amazon Redshift JDBC driver, version 2.x
- Getting the JDBC URL
- Building the connection URL
- Configuring a JDBC connection with Apache Maven
- Configuring authentication and SSL
- Configuring logging
- Data type conversions
- Using prepared statement support
- Differences between the 2.x and 1.x versions of the JDBC driver
- Creating initialization (.ini) files for JDBC driver version 2.x
- Options for JDBC driver version 2.x configuration
- Previous versions of JDBC driver version 2.x

Download the Amazon Redshift JDBC driver, version 2.x

> **Note**
>
> The Amazon Redshift JDBC 2.x driver isn't designed to be thread-safe. Two or more threads concurrently attempting to use the same connection can lead to deadlocks, errors, incorrect results, or other unexpected behaviors.
> If you do have a multi-threaded application, we recommend that you synchronize access to the driver to avoid concurrent access.

Amazon Redshift offers drivers for tools that are compatible with the JDBC 4.2 API. The class name for this driver is `com.amazon.redshift.Driver`.

For detailed information about how to install the JDBC driver, reference the JDBC driver libraries, and register the driver class, see the following topics.

For each computer where you use the Amazon Redshift JDBC driver version 2.x, make sure that the Java Runtime Environment (JRE) 8.0 is installed.

If you use the Amazon Redshift JDBC driver for database authentication, make sure that you have AWS SDK for Java 2.40.5 or later in your Java class path. If you don't have AWS SDK for Java installed, download the ZIP file with JDBC 4.2–compatible driver and driver dependent libraries for the AWS SDK:

- JDBC 4.2–compatible driver version 2.x and AWS SDK driver–dependent libraries

  This ZIP file contains the JDBC 4.2–compatible driver version 2.x and AWS SDK for Java 2.x driver–dependent library files. Unzip the dependent jar files to the same location as the JDBC driver. Only the JDBC driver needs to be in CLASSPATH.

  This ZIP file doesn't include the complete AWS SDK for Java 2.x. However, it includes the AWS SDK for Java 2.x driver–dependent libraries that are required for AWS Identity and Access Management (IAM) database authentication.

  Use this Amazon Redshift JDBC driver with the AWS SDK that is required for IAM database authentication.

  To install the complete AWS SDK for Java 2.x, see AWS SDK for Java 2.x in the *AWS SDK for Java Developer Guide*.

- JDBC 4.2–compatible driver version 2.x (without the AWS SDK)

Review the JDBC driver version 2.x software license and change log file:

- JDBC driver version 2.x license
- JDBC driver version 2.x change log

JDBC drivers version 1.2.27.1051 and later support Amazon Redshift stored procedures. For more information, see Creating stored procedures in Amazon Redshift in the *Amazon Redshift Database Developer Guide*.

Installing the Amazon Redshift JDBC driver, version 2.x

To install the Amazon Redshift JDBC 4.2–compatible driver version 2.x and driver–dependent libraries for AWS SDK, extract the files from the ZIP archive to the directory of your choice.

To install the Amazon Redshift JDBC 4.2–compatible driver version 2.x (without the AWS SDK), copy the JAR file to the directory of your choice.

To access an Amazon Redshift data store using the Amazon Redshift JDBC driver, you need to perform configuration as described following.

Topics

- Referencing the JDBC driver libraries
- Registering the driver class

Referencing the JDBC driver libraries

The JDBC application or Java code that you use to connect to your data must access the driver JAR files. In the application or code, specify all the JAR files that you extracted from the ZIP archive.

Using the driver in a JDBC application

JDBC applications usually provide a set of configuration options for adding a list of driver library files. Use the provided options to include all the JAR files from the ZIP archive as part of the driver configuration in the application. For more information, see the documentation for your JDBC application.

Using the driver in Java code

You must include all the driver library files in the class path. This is the path that the Java Runtime Environment searches for classes and other resource files. For more information, see the appropriate Java SE documentation to set the class path for your operating system.

- Windows: https://docs.oracle.com/javase/7/docs/technotes/tools/windows/classpath.html
- Linux and Solaris: https://docs.oracle.com/javase/7/docs/technotes/tools/solaris/classpath.html
- macOS: The default macOS class path is the directory in which the JDBC driver is installed.

Registering the driver class

Make sure that you register the appropriate class for your application. You use following classes to connect the Amazon Redshift JDBC driver to Amazon Redshift data stores:

- `Driver` classes extend `java.sql.Driver`.
- `DataSource` classes extend `javax.sql.DataSource` and `javax.sql.ConnectionPoolDataSource`.

The driver supports the following fully qualified class names that are independent of the JDBC version:

- `com.amazon.redshift.jdbc.Driver`
- `com.amazon.redshift.jdbc.DataSource`

The following example shows how to use the DriverManager class to establish a connection for JDBC 4.2.

```
            private static Connection connectViaDM() throws Exception
{
Connection connection = null;
connection = DriverManager.getConnection(CONNECTION_URL);
return connection;
}
```

The following example shows how to use the `DataSource` class to establish a connection.

```
 private static Connection connectViaDS() throws Exception
{
Connection connection = null;
11
Amazon Redshift JDBC Driver Installation and Configuration Guide
DataSource ds = new com.amazon.redshift.jdbc.DataSource
();
ds.setURL(CONNECTION_URL);
connection = ds.getConnection();
return connection;
}
```

Getting the JDBC URL

Before you can connect to your Amazon Redshift cluster from a SQL client tool, you need to know the JDBC URL of your cluster. The JDBC URL has the following format: `jdbc:redshift://`*`endpoint`*`:`*`port`*`/`*`database`*.

The fields of the preceding format have the following values.

Field	Value
`jdbc`	The protocol for the connection.
`redshift`	The subprotocol that specifies to use the Amazon Redshift driver to connect to the database.
`endpoint`	The endpoint of the Amazon Redshift cluster.
`port`	The port number that you specified when you launched the cluster. If you have a firewall, make sure that this port is open for you to use.
`database`	The database that you created for your cluster.

The following is an example JDBC URL: `jdbc:redshift://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/dev`

If your URL values contain any of the following URI reserved characters, the values must be URL encoded:

- ;
- +
- {
- }
- [
-]
- &
- =
- ?
- an empty space

For example, if your PWD value is `password:password`, a connection URL using that value would look something like the following:

```
jdbc:redshift://redshift.company.us-west-1.redshift.amazonaws.com:9000/
dev;UID=amazon;PWD=password%3Apassword
```

For information about how to get your JDBC connection, see Finding your cluster connection string.

If the client computer fails to connect to the database, you can troubleshoot possible issues. For more information, see Troubleshooting connection issues in Amazon Redshift.

Building the connection URL

Use the connection URL to supply connection information to the data store that you are accessing. The following is the format of the connection URL for the Amazon Redshift JDBC driver version 2.x. Here, [Host] the endpoint of the Amazon Redshift server and [Port] is the number of the Transmission Control Protocol (TCP) port that the server uses to listen for client requests.

```
jdbc:redshift://[Host]:[Port]
```

The following is the format of a connection URL that specifies some optional settings.

```
jdbc:redshift://[Host]:[Port]/[database];[Property1]=[Value];
[Property2]=[Value];
```

If your URL values contain any of the following URI reserved characters, the values must be URL encoded:

- ;
- +
- {
- }
- [
-]
- &
- =
- ?
- an empty space

For example, if your PWD value is `password:password`, a connection URL using that value would look something like the following:

`jdbc:redshift://redshift.company.us-west-1.redshift.amazonaws.com:9000/dev;UID=amazon;PWD=password%3Apassword`

For example, suppose that you want to connect to port 9000 on an Amazon Redshift cluster in the US West (N. California) Region on AWS. You also want to access the database named `dev` and authenticate the connection using a database username and password. In this case, you use the following connection URL.

```
jdbc:redshift://redshift.company.us-west-1.redshift.amazonaws.com:9000/
dev;UID=amazon;PWD=amazon
```

You can use the following characters to separate the configuration options from the rest of the URL string:

- ;

- ?

For example, the following URL strings are equivalent:

```
jdbc:redshift://my_host:5439/dev;ssl=true;defaultRowFetchSize=100
```

```
jdbc:redshift://my_host:5439/dev?ssl=true;defaultRowFetchSize=100
```

You can use the following characters to separate configuration options from each other in the URL string:

- ;
- &

For example, the following URL strings are equivalent:

```
jdbc:redshift://my_host:5439/dev;ssl=true;defaultRowFetchSize=100
```

```
jdbc:redshift://my_host:5439/dev;ssl=true&defaultRowFetchSize=100
```

The following URL example specifies a log level of 6 and the path for the logs.

```
jdbc:redshift://redshift.amazonaws.com:5439/dev;DSILogLevel=6;LogPath=/home/user/logs;
```

Don't duplicate properties in the connection URL.

For a complete list of the configuration options that you can specify, see Options for JDBC driver version 2.x configuration.

Note

When you connect, don't use the IP address of a cluster node or the IP address of the VPC endpoint. Always use the Redshift endpoint to avoid an unnecessary outage. The only exception to using the endpoint URL is when you use a custom domain name. For more information, see Using a custom domain name for client connections.

Configuring a JDBC connection with Apache Maven

Apache Maven is a software project management and comprehension tool. The AWS SDK for Java supports Apache Maven projects. For more information, see Using the SDK with Apache Maven in the *AWS SDK for Java Developer Guide.*

If you use Apache Maven, you can configure and build your projects to use an Amazon Redshift JDBC driver to connect to your Amazon Redshift cluster. To do this, add the JDBC driver as a dependency in your project's `pom.xml` file. If you use Maven to build your project and want to use a JDBC connection, take the steps in the following section.

To configure the JDBC driver as a Maven dependency

1. Add either the Amazon repository or the Maven Central repository to the repositories section of your `pom.xml` file.

   > **Note**
   >
   > The URL in the following code example returns an error if used in a browser. Use this URL only in the context of a Maven project.

   To connect using Secure Sockets Layer (SSL), add the following repository to your `pom.xml` file.

   ```
   <repositories>
       <repository>
         <id>redshift</id>
         <url>https://s3.amazonaws.com/redshift-maven-repository/release</url>
       </repository>
   </repositories>
   ```

   For a Maven Central repository, add the following to your `pom.xml` file.

   ```
   <repositories>
       <repository>
         <id>redshift</id>
         <url>https://repo1.maven.org/maven2</url>
       </repository>
   </repositories>
   ```

2. Declare the version of the driver that you want to use in the dependencies section of your `pom.xml` file.

   Amazon Redshift offers drivers for tools that are compatible with the JDBC 4.2 API. For information about the functionality supported by these drivers, see Download the Amazon Redshift JDBC driver, version 2.x.

   Replace *`driver-version`* in the following example with your driver version, for example `2.1.0.1`. For a JDBC 4.2–compatible driver, use the following.

   ```
   <dependency>
      <groupId>com.amazon.redshift</groupId>
      <artifactId>redshift-jdbc42</artifactId>
      <version>driver-version</version>
   </dependency>
   ```

   The class name for this driver is `com.amazon.redshift.Driver`.

The Amazon Redshift Maven drivers need the following optional dependencies when you use IAM database authentication.

```
<dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-core</artifactId>
      <version>1.12.23</version>
      <scope>runtime</scope>
      <optional>true</optional>
</dependency>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-redshift</artifactId>
      <version>1.12.23</version>
      <scope>runtime</scope>
      <optional>true</optional>
</dependency>
<dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-sts</artifactId>
      <version>1.12.23</version>
      <scope>runtime</scope>
      <optional>true</optional>
```

```
</dependency>
```

To upgrade or change the Amazon Redshift JDBC driver to the latest version, first modify the version section of the dependency to the latest version of the driver. Then clean your project with the Maven Clean Plugin, as shown following.

```
mvn clean
```

Configuring authentication and SSL

To protect data from unauthorized access, Amazon Redshift data stores require all connections to be authenticated using user credentials. Some data stores also require connections to be made over the Secure Sockets Layer (SSL) protocol, either with or without one-way authentication.

The Amazon Redshift JDBC driver version 2.x provides full support for these authentication protocols.

The SSL version that the driver supports depends on the JVM version that you are using. For information about the SSL versions that are supported by each version of Java, see Diagnosing TLS, SSL, and HTTPS on the Java Platform Group Product Management Blog.

The SSL version used for the connection is the highest version that is supported by both the driver and the server, which is determined at connection time.

Configure the Amazon Redshift JDBC driver version 2.x to authenticate your connection according to the security requirements of the Redshift server that you are connecting to.

You must always provide your Redshift username and password to authenticate the connection. Depending on whether SSL is enabled and required on the server, you might also need to configure the driver to connect through SSL. Or you might use one-way SSL authentication so that the client (the driver itself) verifies the identity of the server.

You provide the configuration information to the driver in the connection URL. For more information about the syntax of the connection URL, see Building the connection URL.

SSL indicates TLS/SSL, both Transport Layer Security and Secure Sockets Layer. The driver supports industry-standard versions of TLS/SSL.

Configuring IAM authentication

If you are connecting to a Amazon Redshift server using IAM authentication, set the following properties as part of your data source connection string.

For more information on IAM authentication, see Identity and access management in Amazon Redshift.

To use IAM authentication, use one of the following connection string formats:

Connection string	Description
`jdbc:redshift:iam:// [host]:[port]/[db]`	A regular connection string. The driver infers the ClusterID and Region from the host.
`jdbc:redshift:iam:// [cluster-id]: [region]/[db]`	The driver retrieves host information, given the ClusterID and Region.
`jdbc:redshift:iam:// [host]/[db]`	The driver defaults to port 5439, and infers ClusterID and Region from the host. Depending on the port you selected when creating, modifying or migrating the cluster, allow access to the selected port.

Specifying profiles

If you are using IAM authentication, you can specify any additional required or optional connection properties under a profile name. By doing this, you can avoid putting certain information directly in the connection string. You specify the profile name in your connection string using the Profile property.

Profiles can be added to the AWS credentials file. The default location for this file is: `~/.aws/credentials`

You can change the default value by setting the path in the following environment variable: `AWS_CREDENTIAL_PROFILES_FILE`

For more information about profiles, see Working with AWS Credentials in the *AWS SDK for Java*.

Using instance profile credentials

If you are running an application on an Amazon EC2 instance that is associated with an IAM role, you can connect using the instance profile credentials.

To do this, use one of the IAM connection string formats in the preceding table, and set the dbuser connection property to the Amazon Redshift username that you are connecting as.

For more information about instance profiles, see Access Management in the *IAM User Guide*.

Using credential providers

The driver also supports credential provider plugins from the following services:

- AWS IAM Identity Center
- Active Directory Federation Service (ADFS)
- JSON Web Tokens (JWT) Service
- Microsoft Azure Active Directory (AD) Service and Browser Microsoft Azure Active Directory (AD) Service
- Okta Service
- PingFederate Service
- Browser SAML for SAML services such as Okta, Ping, or ADFS

If you use one of these services, the connection URL needs to specify the following properties:

- **Plugin_Name** – The fully-qualified class path for your credentials provider plugin class.
- **IdP_Host:** – The host for the service that you are using to authenticate into Amazon Redshift.
- **IdP_Port** – The port that the host for the authentication service listens at. Not required for Okta.
- **User** – The username for the idp_host server.
- **Password** – The password associated with the idp_host username.
- **DbUser** – The Amazon Redshift username you are connecting as.
- **SSL_Insecure** – Indicates whether the IDP server certificate should be verified.
- **Client_ID** – The client ID associated with the username in the Azure AD portal. Only used for Azure AD.
- **Client_Secret** – The client secret associated with the client ID in the Azure AD portal. Only used for Azure AD.
- **IdP_Tenant** – The Azure AD tenant ID for your Amazon Redshift application. Only used for Azure AD.
- **App_ID** – The Okta app ID for your Amazon Redshift application. Only used for Okta.

- **App_Name** – The optional Okta app name for your Amazon Redshift application. Only used for Okta.
- **Partner_SPID** – The optional partner SPID (service provider ID) value. Only used for PingFederate.
- **Idc_Region** – The AWS Region where the AWS IAM Identity Center instance is located. Only used for AWS IAM Identity Center.
- **Issuer_Url** – The AWS IAM Identity Center server's instance endpoint. Only used for AWS IAM Identity Center.

If you are using a browser plugin for one of these services, the connection URL can also include:

- **Login_URL** –The URL for the resource on the identity provider's website when using the Security Assertion Markup Language (SAML) or Azure AD services through a browser plugin. This parameter is required if you are using a browser plugin.

- **Listen_Port** – The port that the driver uses to get the SAML response from the identity provider when using the SAML, Azure AD, or AWS IAM Identity Center services through a browser plugin.

- **IdP_Response_Timeout** – The amount of time, in seconds, that the driver waits for the SAML response from the identity provider when using the SAML, Azure AD, or AWS IAM Identity Center services through a browser plugin.

For information on additional connection string properties, see Options for JDBC driver version 2.x configuration.

Using username and password only

If the server you are connecting to doesn't use SSL, then you only need to provide your Redshift username and password to authenticate the connection.

To configure authentication using your Redshift username and password only

1. Set the `UID` property to your Redshift username for accessing the Amazon Redshift server.
2. Set the PWD property to the password corresponding to your Redshift username.

Using SSL without identity verification

If the server you are connecting to uses SSL but doesn't require identity verification, then you can configure the driver to use a non-validating SSL factory.

To configure an SSL connection without identity verification

1. Set the `UID` property to your Redshift username for accessing the Amazon Redshift server.
2. Set the `PWD` property to the password corresponding to your Redshift username.
3. Set the `SSLFactory` property to `com.amazon.redshift.ssl.NonValidatingFactory`.

Using one-way SSL authentication

If the server you are connecting to uses SSL and has a certificate, then you can configure the driver to verify the identity of the server using one-way authentication.

One-way authentication requires a signed, trusted SSL certificate for verifying the identity of the server. You can configure the driver to use a specific certificate or access a TrustStore that contains the appropriate certificate. If you don't specify a certificate or TrustStore, then the driver uses the default Java TrustStore (typically either `jssecacerts` or `cacerts`).

To configure one-way SSL authentication

1. Set the UID property to your Redshift username for accessing the Amazon Redshift server.
2. Set the PWD property to the password corresponding to your Redshift username.
3. Set the SSL property to true.
4. Set the SSLRootCert property to the location of your root CA certificate.
5. If you aren't using one of the default Java TrustStores, then do one of the following:
   - To specify a server certificate, set the SSLRootCert property to the full path of the certificate.
   - To specify a TrustStore, do the following:

     a. Use the keytool program to add the server certificate to the TrustStore that you want to use.

     b. Specify the TrustStore and password to use when starting the Java application using the driver. For example:

```
-Djavax.net.ssl.trustStore=[TrustStoreName]
```

```
-Djavax.net.ssl.trustStorePassword=[TrustStorePassword]
-Djavax.net.ssl.trustStoreType=[TrustStoreType]
```

6. Choose one:
   - To validate the certificate, set the SSLMode property to verify-ca.
   - To validate the certificate and verify the host name in the certificate, set the SSLMode property to verify-full.

Configuring logging

You can turn on logging in the driver to assist in diagnosing issues.

You can log driver information by using the following methods:

- To save logged information in .log files, see Using log files.
- To send logged information to the LogStream or LogWriter specified in the DriverManager, see Using LogStream or LogWriter.

You provide the configuration information to the driver in the connection URL. For more information about the syntax of the connection URL, see Building the connection URL.

Using log files

Only turn on logging long enough to capture an issue. Logging decreases performance and can consume a large quantity of disk space.

Set the LogLevel key in your connection URL to turn on logging and specify the amount of detail included in log files. The following table lists the logging levels provided by the Amazon Redshift JDBC driver version 2.x, in order from least verbose to most verbose.

LogLevel value	Description
1	Log severe error events that will lead the driver to abort.
2	Log error events that might allow the driver to continue running.

LogLevel value	Description
3	Log events that might result in an error if action is not taken. This level of logging and the levels of logging above this level also log the user's queries.
4	Log general information that describes the progress of the driver.
5	Log detailed information that is useful for debugging the driver.
6	Log all driver activity.

To set up logging that uses log files

1. Set the LogLevel property to the desired level of information to include in log files.
2. Set the LogPath property to the full path to the folder where you want to save log files.

   For example, the following connection URL enables logging level 3 and saves the log files in the C:\temp folder: `jdbc:redshift://redshift.company.us-west-1.redshift.amazonaws.com:9000/Default;DSILogLevel=3;LogPath=C:\temp`
3. To make sure that the new settings take effect, restart your JDBC application and reconnect to the server.

   The Amazon Redshift JDBC driver produces the following log files in the location specified in the LogPath property:

   - redshift_jdbc.log file that logs driver activity that is not specific to a connection.
   - redshift_jdbc_connection_[Number].log file for each connection made to the database, where [Number] is a number that identifies each log file. This file logs driver activity that is specific to the connection.

If the LogPath value is invalid, then the driver sends the logged information to the standard output stream (`System.out`)

Using LogStream or LogWriter

Only turn on logging long enough to capture an issue. Logging decreases performance and can consume a large quantity of disk space.

Set the LogLevel key in your connection URL to turn on logging and specify the amount of detail sent to the LogStream or LogWriter specified in the DriverManager.

To turn on logging that uses the LogStream or LogWriter:

1. To configure the driver to log general information that describes the progress of the driver, set the LogLevel property to 1 or INFO.
2. To make sure that the new settings take effect, restart your JDBC application and reconnect to the server.

Data type conversions

The Amazon Redshift JDBC driver version 2.x supports many common data formats, converting between Amazon Redshift, SQL, and Java data types.

The following table lists the supported data type mappings.

Amazon Redshift type	SQL type	Java type
BIGINT	SQL_BIGINT	Long
BOOLEAN	SQL_BIT	Boolean
CHAR	SQL_CHAR	String
DATE	SQL_TYPE_DATE	java.sql.Date
DECIMAL	SQL_NUMERIC	BigDecimal
DOUBLE PRECISION	SQL_DOUBLE	Double
GEOMETRY	SQL_ LONGVARBINARY	byte[]
INTEGER	SQL_INTEGER	Integer
OID	SQL_BIGINT	Long

Amazon Redshift type	SQL type	Java type
SUPER	SQL_LONGVARCHAR	String
REAL	SQL_REAL	Float
SMALLINT	SQL_SMALLINT	Short
TEXT	SQL_VARCHAR	String
TIME	SQL_TYPE_TIME	java.sql.Time
TIMETZ	SQL_TYPE_TIME	java.sql.Time
TIMESTAMP	SQL_TYPE_ TIMESTAMP	java.sql.Timestamp
TIMESTAMPTZ	SQL_TYPE_ TIMESTAMP	java.sql.Timestamp
VARCHAR	SQL_VARCHAR	String

Using prepared statement support

The Amazon Redshift JDBC driver supports prepared statements. You can use prepared statements to improve the performance of parameterized queries that need to be run multiple times during the same connection.

A *prepared statement* is a SQL statement that is compiled on the server side but not run immediately. The compiled statement is stored on the server as a PreparedStatement object until you close the object or the connection. While that object exists, you can run the prepared statement as many times as needed using different parameter values, without having to compile the statement again. This reduced overhead enables the set of queries to be run more quickly.

For more information about prepared statements, see "Using Prepared Statements" in JDBC Basics tutorial from Oracle.

You can prepare a statement that contains multiple queries. For example, the following prepared statement contains two INSERT queries:

```
PreparedStatement pstmt = conn.prepareStatement("INSERT INTO
MyTable VALUES (1, 'abc'); INSERT INTO CompanyTable VALUES
```

```
(1, 'abc');");
```

Take care that these queries don't depend on the results of other queries that are specified within the same prepared statement. Because queries don't run during the prepare step, the results have not been returned yet, and aren't available to other queries in the same prepared statement.

For example, the following prepared statement, which creates a table and then inserts values into that newly-created table, is not allowed:

```
PreparedStatement pstmt = conn.prepareStatement("CREATE
TABLE MyTable(col1 int, col2 varchar); INSERT INTO myTable
VALUES (1, 'abc');");
```

If you try to prepare this statement, the server returns an error stating that the destination table (myTable) doesn't exist yet. The CREATE query must be run before the INSERT query can be prepared.

Differences between the 2.x and 1.x versions of the JDBC driver

This section describes the differences in the information returned by the 2.x and 1.x versions of the JDBC driver. The JDBC driver version 1.x is discontinued.

The following table lists the DatabaseMetadata information returned by the getDatabaseProductName() and getDatabaseProductVersion() functions for each version of the JDBC driver. JDBC driver version 2.x obtains the values while establishing the connection. JDBC driver version 1.x obtains the values as a result of a query.

JDBC driver version	getDatabaseProductName() result	getDatabaseProduct Version() result
2.x	Redshift	8.0.2
1.x	PostgreSQL	08.00.0002

The following table lists the DatabaseMetadata information returned by the getTypeInfo function for each version of the JDBC driver.

JDBC driver version	getTypeInfo result
2.x	Consistent with Redshift datatypes
1.x	Consistent with PostgreSQL datatypes

Creating initialization (.ini) files for JDBC driver version 2.x

By using initialization (.ini) files for Amazon Redshift JDBC driver version 2.x, you can specify system level configuration parameters. For example, federated IdP authentication parameters can vary for each application. The .ini file provides a common location for SQL clients to get the required configuration parameters.

You can create an JDBC driver version 2.x initialization (.ini) file that contains configuration options for SQL clients. The default name of the file is `rsjdbc.ini`. The JDBC driver version 2.x checks for the .ini file in the following locations, listed in order of precedence:

- `IniFile` parameter in the connection URL or in the connection property dialog box of the SQL client. Be sure that the `IniFile` parameter contains the full path to the .ini file, including the file name. For information about the `IniFile` parameter, see IniFile. If the `IniFile` parameter incorrectly specifies the location of the .ini file, an error displays.
- Environment variables such as AMAZON_REDSHIFT_JDBC_INI_FILE with the full path, including the file name. You can use `rsjdbc.ini` or specify a file name. If the AMAZON_REDSHIFT_JDBC_INI_FILE environment variable incorrectly specifies the location of the .ini file, an error displays.
- Directory where the driver JAR file is located.
- User home directory.
- Temp directory of the system.

You can organize the .ini file into sections, for example [DRIVER]. Each section contains key-value pairs that specify various connection parameters. You can use the `IniSection` parameter to specify a section in the .ini file. For information about the `IniSection` parameter, see IniSection.

Following is an example of the .ini file format, with sections for [DRIVER], [DEV], [QA], and [PROD]. The [DRIVER] section can apply to any connection.

```
[DRIVER]
```

```
key1=val1
key2=val2

[DEV]
key1=val1
key2=val2

[QA]
key1=val1
key2=val2

[PROD]
key1=val1
key2=val2
```

The JDBC driver version 2.x loads configuration parameters from the following locations, listed in order of precedence:

- Default configuration parameters in the application code.
- [DRIVER] section properties from the .ini file, if included.
- Custom section configuration parameters, if the `IniSection` option is provided in the connection URL or in the connection property dialog box of the SQL client.
- Properties from the connection property object specified in the `getConnection` call.
- Configuration parameters speified in the connection URL.

Options for JDBC driver version 2.x configuration

Following, you can find descriptions for the options that you can specify for version 2.x of the Amazon Redshift JDBC driver. Configuration options are not case sensitive.

You can set configuration properties using the connection URL. For more information, see Building the connection URL.

Topics

- AccessKeyID
- AllowDBUserOverride
- App_ID
- App_Name

- ApplicationName
- AuthProfile
- AutoCreate
- Client_ID
- Client_Secret
- ClusterID
- Compression
- connectTimeout
- connectionTimezone
- databaseMetadataCurrentDbOnly
- DbUser
- DbGroups
- DBNAME
- defaultRowFetchSize
- DisableIsValidQuery
- enableFetchRingBuffer
- enableMultiSqlSupport
- EnableTableTypes
- fetchRingBufferSize
- ForceLowercase
- groupFederation
- HOST
- IAMDisableCache
- IAMDuration
- Idc_Client_Display_Name
- Idc_Region
- IdP_Host
- IdP_Partition
- IdP_Port

- IdP_Tenant
- IdP_Response_Timeout
- IniFile
- IniSection
- isServerless
- Issuer_Url
- Listen_Port
- Login_URL
- loginTimeout
- loginToRp
- LogLevel
- LogPath
- OverrideSchemaPatternType
- Partner_SPID
- Password
- Plugin_Name
- PORT
- preferPQ
- Preferred_Role
- Profile
- PWD
- queryGroup
- readOnly
- Region
- reWriteBatchedInserts
- reWriteBatchedInsertsSize
- roleArn
- roleSessionName
- scope

- SecretAccessKey
- SessionToken
- serverlessAcctId
- serverlessWorkGroup
- socketFactory
- socketTimeout
- SSL
- SSL_Insecure
- SSLCert
- SSLFactory
- SSLKey
- SSLMode
- SSLPassword
- SSLRootCert
- StsEndpointUrl
- tcpKeepAlive
- token
- token_type
- UID
- User
- webIdentityToken

AccessKeyID

- **Default Value** – None
- **Data Type** – String

You can specify this parameter to enter the IAM access key for the user or role. You can usually locate the key by looking at and existing string or user profile. If you specify this parameter, you must also specify the `SecretAccessKey` parameter. If passed in the JDBC URL, AccessKeyID must be URL encoded.

This parameter is optional.

AllowDBUserOverride

- **Default Value** – 0
- **Data Type** – String

This option specifies whether the driver uses the `DbUser` value from the SAML assertion or the value that is specified in the `DbUser` connection property in the connection URL.

This parameter is optional.

1

The driver uses the `DbUser` value from the SAML assertion.

If the SAML assertion doesn't specify a value for `DBUser`, the driver uses the value specified in the `DBUser` connection property. If the connection property also doesn't specify a value, the driver uses the value specified in the connection profile.

0

The driver uses the `DBUser` value specified in the `DBUser` connection property.

If the `DBUser` connection property doesn't specify a value, the driver uses the value specified in the connection profile. If the connection profile also doesn't specify a value, the driver uses the value from the SAML assertion.

App_ID

- **Default Value** – None
- **Data Type** – String

The Okta-provided unique ID associated with your Amazon Redshift application.

This parameter is required if authenticating through the Okta service.

App_Name

- **Default Value** – None

- **Data Type** – String

The name of the Okta application that you use to authenticate the connection to Amazon Redshift.

This parameter is optional.

ApplicationName

- **Default Value** – null
- **Data Type** – String

The name of the application to pass to Amazon Redshift for audit purposes.

This parameter is optional.

AuthProfile

- **Default Value** – None
- **Data Type** – String

The name of the authentication profile to use for connecting to Amazon Redshift.

This parameter is optional.

AutoCreate

- **Default Value** – false
- **Data Type** – Boolean

This option specifies whether the driver causes a new user to be created when the specified user doesn't exist.

This parameter is optional.

true

If the user specified by either `DBUser` or unique ID (UID) doesn't exist, a new user with that name is created.

false

The driver doesn't cause new users to be created. If the specified user doesn't exist, the authentication fails.

Client_ID

- **Default Value** – None
- **Data Type** – String

The client ID to use when authenticating the connection using the Azure AD service.

This parameter is required if authenticating through the Azure AD service.

Client_Secret

- **Default Value** – None
- **Data Type** – String

The Client Secret to use when authenticating the connection using the Azure AD service.

This parameter is required if authenticating through the Azure AD service.

ClusterID

- **Default Value** – None
- **Data Type** – String

The name of the Amazon Redshift cluster that you want to connect to. The driver attempts to detect this parameter from the given host. If you're using a Network Load Balancer (NLB) and connecting via IAM, the driver will fail to detect it, so you can set it using this connection option.

This parameter is optional.

Compression

- **Default Value** – off
- **Data Type** – String

The compression method used for wire protocol communication between the Amazon Redshift server and the client or driver.

This parameter is optional.

You can specify the following values:

- **lz4**

  Sets the compression method used for wire protocol communication with Amazon Redshift to lz4.
- **off**

  Doesn't use compression for wire protocol communication with Amazon Redshift.

connectTimeout

- **Default Value** – 10
- **Data Type** – Integer

The timeout value to use for socket connect operations. If the time required to establish an Amazon Redshift connection exceeds this value, the connection is considered unavailable. The timeout is specified in seconds. A value of 0 means that no timeout is specified.

This parameter is optional.

connectionTimezone

- **Default Value** – LOCAL
- **Data Type** – String

The session level timezone.

This parameter is optional.

You can specify the following values:

LOCAL

Configures the session level timezone to the LOCAL JVM timezone.

SERVER

Configures the session level timezone to the timezone set for the user on the Amazon Redshift server. You can configure session level timezones for users with the following command:

```
ALTER USER
[...]
SET TIMEZONE TO [...];
```

databaseMetadataCurrentDbOnly

- **Default Value** – true
- **Data Type** – Boolean

This option specifies whether the metadata API retrieves data from all accessible databases or only from the connected database.

This parameter is optional.

You can specify the following values:

true

The application retrieves metadata from a single database.

false

The application retrieves metadata from all accessible databases.

DbUser

- **Default Value** – None
- **Data Type** – String

The user ID to use with your Amazon Redshift account. You can use an ID that doesn't currently exist if you have enabled the AutoCreate property.

This parameter is optional.

DbGroups

- **Default Value** – PUBLIC
- **Data Type** – String

A comma-separated list of existing database group names that `DBUser` joins for the current session.

This parameter is optional.

DBNAME

- **Default Value** – null
- **Data Type** – String

The name of the database to connect to. You can use this option to specify the database name in the JDBC connection URL.

This parameter is required. You must specify the database name, either in the connection URL or in the connection properties of the client application.

defaultRowFetchSize

- **Default Value** – 0
- **Data Type** – Integer

This option specifies a default value for getFetchSize.

This parameter is optional.

You can specify the following values:

0

Fetch all rows in a single operation.

Positive integer

Number of rows to fetch from the database for each fetch iteration of the ResultSet.

DisableIsValidQuery

- **Default Value** – False
- **Data Type** – Boolean

This option specifies whether the driver submits a new database query when using the Connection.isValid() method to determine whether the database connection is active.

This parameter is optional.

true

The driver doesn't submit a query when using Connection.isValid() to determine whether the database connection is active. This may cause the driver to incorrectly identify the database connection as active if the database server has shut down unexpectedly.

false

The driver submits a query when using Connection.isValid() to determine whether the database connection is active.

enableFetchRingBuffer

- **Default Value** – true
- **Data Type** – Boolean

This option specifies that the driver fetches rows using a ring buffer on a separate thread. The fetchRingBufferSize parameter specifies the ring buffer size.

The ring buffer implements automatic memory management in JDBC to prevent out-of-memory (OOM) errors during data retrieval operations. The ring buffer monitors the actual size of buffered data in real-time, ensuring total memory usage by the driver stays within defined limits. When buffer capacity is reached, the driver pauses data fetching operations, preventing memory overflow without requiring manual intervention. This built-in safeguard eliminates OOM errors automatically, with no configuration needed from users.

If a transaction detects a Statement containing multiple SQL commands separated by semicolons, the fetch ring buffer for that transaction is set to false. enableFetchRingBuffer's value doesn't change.

This parameter is optional.

> **Note**
>
> When the ring buffer is disabled and the fetch size is not properly configured, out-of-memory (OOM) issues may occur. For more information about configuring fetch size, see here.

enableMultiSqlSupport

- **Default Value** – true
- **Data Type** – Boolean

This option specifies whether to process multiple SQL commands separated by semicolons in a Statement.

This parameter is optional.

You can specify the following values:

true

The driver processes multiple SQL commands, separated by semicolons, in a Statement object.

false

The driver returns an error for multiple SQL commands in a single Statement.

EnableTableTypes

- **Default Value** – true
- **Data Type** – Boolean

This option specifies whether the driver recognizes detailed table type information from the data source in the results of the `getTables` and `getTableTypes` `DatabaseMetaData` methods. By default, the driver recognizes detailed table types.

This parameter is optional. It is available in driver versions 2.2.8 and later.

true

The driver recognizes the following table types: TABLE, VIEW, SYSTEM TABLE, SYSTEM VIEW, EXTERNAL TABLE, and LOCAL TEMPORARY.

false

The driver normalizes the detailed table type information into the generic TABLE and VIEW table types.

fetchRingBufferSize

- **Default Value** – 1G
- **Data Type** – String

This option specifies the size of the ring buffer used while fetching the result set. You can specify a size in bytes, for example 1K for 1 KB, 5000 for 5,000 bytes, 1M for 1 MB, 1G for 1 GB, and so on. You can also specify a percentage of heap memory. The driver stops fetching rows upon reaching the limit. Fetching resumes when the application reads rows and frees space in the ring buffer.

This parameter is optional.

ForceLowercase

- **Default Value** – false
- **Data Type** – Boolean

This option specifies whether the driver lowercases all database groups (DbGroups) sent from the identity provider to Amazon Redshift when using single sign-on authentication.

This parameter is optional.

true

The driver lowercases all database groups that are sent from the identity provider.

false

The driver doesn't alter database groups.

groupFederation

- **Default Value** – false
- **Data Type** – Boolean

This option specifies whether to use Amazon Redshift IDP groups. This is supported by the GetClusterCredentialsV2 API.

This parameter is optional.

true

Use Amazon Redshift Identity Provider (IDP) groups.

false

Use STS API and GetClusterCredentials for user federation and explicitly specify DbGroups for the connection.

HOST

- **Default Value** – null
- **Data Type** – String

The host name of the Amazon Redshift server to connect to. You can use this option to specify the host name in the JDBC connection URL.

This parameter is required. You must specify the host name, either in the connection URL or in the connection properties of the client application.

IAMDisableCache

- **Default Value** – false
- **Data Type** – Boolean

This option specifies whether the IAM credentials are cached.

This parameter is optional.

true

The IAM credentials aren't cached.

false

The IAM credentials are cached. This improves performance when requests to the API gateway are throttled, for instance.

IAMDuration

- **Default Value** – 900
- **Data Type** – Integer

The length of time, in seconds, until the temporary IAM credentials expire.

- **Minimum value** – 900
- **Maximum value** – 3,600

This parameter is optional.

Idc_Client_Display_Name

- **Default Value** – Amazon Redshift JDBC driver
- **Data Type** – String

The display name to be used for the client that's using BrowserIdcAuthPlugin.

This parameter is optional.

Idc_Region

- **Default Value** – None
- **Data Type** – String

The AWS region where the IAM Identity Center instance is located.

This parameter is required only when authenticating using `BrowserIdcAuthPlugin` in the plugin_name configuration option.

IdP_Host

- **Default Value** – None
- **Data Type** – String

The IdP (identity provider) host you are using to authenticate into Amazon Redshift. This can be specified in either the connection string or in a profile.

This parameter is optional.

IdP_Partition

- **Default Value** – None
- **Data Type** – String

Specifies the cloud partition where your identity provider (IdP) is configured. This determines which IdP authentication endpoint the driver connects to.

If this parameter is left blank, the driver defaults to the commercial partition. Possible values are:

- `us-gov`: Use this value if your IdP is configured in Azure Government. For example, Azure AD Government uses the endpoint `login.microsoftonline.us`.
- `cn`: Use this value if your IdP is configured in the China cloud partition. For example, Azure AD China uses the endpoint `login.chinacloudapi.cn`.

This parameter is optional.

IdP_Port

- **Default Value** – None
- **Data Type** – String

The port used by an IdP (identity provider). You can specify the port in either the connection string or in a profile. The default port is 5439. Depending on the port you selected when creating, modifying or migrating the cluster, allow access to the selected port.

This parameter is optional.

IdP_Tenant

- **Default Value** – None
- **Data Type** – String

The Azure AD tenant ID for your Amazon Redshift application.

This parameter is required if authenticating through the Azure AD service.

IdP_Response_Timeout

- **Default Value** – 120
- **Data Type** – Integer

The amount of time, in seconds, that the driver waits for the SAML response from the identity provider when using the SAML or Azure AD services through a browser plugin.

This parameter is optional.

IniFile

- **Default Value** – None
- **Data Type** – String

The full path of the .ini file, including file name. For example:

```
IniFile="C:\tools\rsjdbc.ini"
```

For information about the .ini file, see Creating initialization (.ini) files for JDBC driver version 2.x.

This parameter is optional.

IniSection

- **Default Value** – None
- **Data Type** – String

The name of a section in the .ini file containing the configuration options. For information about the .ini file, see Creating initialization (.ini) files for JDBC driver version 2.x.

The following example specifies the [Prod] section of the .ini file:

```
IniSection="Prod"
```

This parameter is optional.

isServerless

- **Default Value** – false
- **Data Type** – Boolean

This option specifies whether the Amazon Redshift endpoint host is a serverless instance. The driver attempts to detect this parameter from the given host. If you're using a Network Load Balancer (NLB), the driver will fail to detect it, so you can set it here.

This parameter is optional.

true

The Amazon Redshift endpoint host is a serverless instance.

false

The Amazon Redshift endpoint host is a provisioned cluster.

Issuer_Url

- **Default Value** – None
- **Data Type** – String

Points to the AWS IAM Identity Center server's instance endpoint.

This parameter is required only when authenticating using `BrowserIdcAuthPlugin` in the plugin_name configuration option.

Listen_Port

- **Default Value** – 7890
- **Data Type** – Integer

The port that the driver uses to receive the SAML response from the identity provider or authorization code when using SAML, Azure AD, or AWS Identity Center services through a browser plugin.

This parameter is optional.

Login_URL

- **Default Value** – None
- **Data Type** – String

The URL for the resource on the identity provider's website when using the SAML or Azure AD services through a browser plugin.

This parameter is required if authenticating with the SAML or Azure AD services through a browser plugin.

loginTimeout

- **Default Value** – 0
- **Data Type** – Integer

The number of seconds to wait before timing out when connecting and authenticating to the server. If establishing the connection takes longer than this threshold, then the connection is aborted.

When this property is set to 0, connections don't time out.

This parameter is optional.

loginToRp

- **Default Value** – `urn:amazon:webservices`
- **Data Type** – String

The relying party trust that you want to use for the AD FS authentication type.

This parameter is optional.

LogLevel

- **Default Value** – 0
- **Data Type** – Integer

Use this property to turn on or turn off logging in the driver and to specify the amount of detail included in log files.

Enable logging only long enough to capture an issue. Logging decreases performance and can consume a large quantity of disk space.

This parameter is optional.

Set the parameter to one of the following values:

0

Disable all logging.

1

Enable logging on the FATAL level, which logs very severe error events that will lead the driver to abort.

2

Enable logging on the ERROR level, which logs error events that might still allow the driver to continue running.

3

Enable logging on the WARNING level, which logs events that might result in an error if action is not taken.

4

Enable logging on the INFO level, which logs general information that describes the progress of the driver.

5

Enable logging on the DEBUG level, which logs detailed information that is useful for debugging the driver.

6

Enable logging on the TRACE level, which logs all driver activity.

When logging is enabled, the driver produces the following log files in the location specified in the `LogPath` property:

- **`redshift_jdbc.log`** – File that logs driver activity that is not specific to a connection.
- **`redshift_jdbc_connection_[Number].log`** – File for each connection made to the database, where `[Number]` is a number that distinguishes each log file from the others. This file logs driver activity that is specific to the connection.

If the LogPath value is invalid, the driver sends the logged information to the standard output stream, `System.out`.

LogPath

- **Default Value** – The current working directory.
- **Data Type** – String

The full path to the folder where the driver saves log files when the DSILogLevel property is enabled.

To be sure that the connection URL is compatible with all JDBC applications, we recommend that you escape the backslashes (\) in your file path by typing another backslash.

This parameter is optional.

OverrideSchemaPatternType

- **Default Value** – null
- **Data Type** – Integer

This option specifies whether to override the type of query used in getTables calls.

0

No Schema Universal Query

1

Local Schema Query

2

External Schema Query

This parameter is optional.

Partner_SPID

- **Default Value** – None
- **Data Type** – String

The partner SPID (service provider ID) value to use when authenticating the connection using the PingFederate service.

This parameter is optional.

Password

- **Default Value** – None
- **Data Type** – String

When connecting using IAM authentication through an IDP, this is the password for the IDP_Host server. When using standard authentication, this can be used for the Amazon Redshift database password instead of PWD.

This parameter is optional.

Plugin_Name

- **Default Value** – None
- **Data Type** – String

The fully qualified class name to implement a specific credentials provider plugin.

This parameter is optional.

The following provider options are supported:

- **`AdfsCredentialsProvider`** – Active Directory Federation Service.
- **`AzureCredentialsProvider`** – Microsoft Azure Active Directory (AD) Service.
- **`BasicJwtCredentialsProvider`** – JSON Web Tokens (JWT) Service.
- **`BasicSamlCredentialsProvider`** – Security Assertion Markup Language (SAML) credentials which you can use with many SAML service providers.
- **`BrowserAzureCredentialsProvider`** – Browser Microsoft Azure Active Directory (AD) Service.
- **`BrowserAzureOAuth2CredentialsProvider`** – Browser Microsoft Azure Active Directory (AD) Service for Native Authentication.
- **`BrowserIdcAuthPlugin`** – An authorization plugin using AWS IAM Identity Center.
- **`BrowserSamlCredentialsProvider`** – Browser SAML for SAML services such as Okta, Ping, or ADFS.
- **`IdpTokenAuthPlugin`** – An authorization plugin that accepts an AWS IAM Identity Center token or OpenID Connect (OIDC) JSON-based identity tokens (JWT) from any web identity provider linked to AWS IAM Identity Center.
- **`OktaCredentialsProvider`** – Okta Service.
- **`PingCredentialsProvider`** – PingFederate Service.

PORT

- **Default Value** – null
- **Data Type** – Integer

The port of the Amazon Redshift server to connect to. You can use this option to specify the port in the JDBC connection URL.

This parameter is optional.

preferPQ

- **Default Value** – true
- **Data Type** – Boolean

This option controls whether the driver advertises post-quantum hybrid key-exchange groups in the TLS 1.3 ClientHello. When post-quantum groups are offered and the server also supports them, the server selects one. Otherwise, the handshake falls back to classical key exchange transparently.

This parameter is optional. It is available in driver versions 2.2.8 and later.

true

The driver advertises post-quantum hybrid TLS key-exchange groups when supported by the runtime.

false

The driver does not advertise post-quantum groups. Use this setting if a network middlebox blocks TLS handshakes because it does not recognize the post-quantum key-exchange groups the driver offers.

Preferred_Role

- **Default Value** – None
- **Data Type** – String

The IAM role that you want to assume during the connection to Amazon Redshift.

This parameter is optional.

Profile

- **Default Value** – None
- **Data Type** – String

The name of the profile to use for IAM authentication. This profile contains any additional connection properties not specified in the connection string.

This parameter is optional.

PWD

- **Default Value** – None

- **Data Type** – String

The password corresponding to the Amazon Redshift username that you provided using the property UID.

This parameter is optional.

queryGroup

- **Default Value** – null
- **Data Type** – String

This option assigns a query to a queue at runtime by assigning your query to the appropriate query group. The query group is set for the session. All queries that run on the connection belong to this query group.

This parameter is optional.

readOnly

- **Default Value** – false
- **Data Type** – Boolean

This property specifies whether the driver is in read-only mode.

This parameter is optional.

true

The connection is in read-only mode and cannot write to the data store.

false

The connection is not in read-only mode and can write to the data store.

Region

- **Default Value** – null
- **Data Type** – String

This option specifies the AWS Region where the cluster is located. If you specify the StsEndPoint option, the Region option is ignored. The Redshift `GetClusterCredentials` API operation also uses the Region option.

This parameter is optional.

reWriteBatchedInserts

- **Default Value** – false
- **Data Type** – Boolean

This option enables optimization to rewrite and combine compatible INSERT statements into batches.

This parameter is optional.

reWriteBatchedInsertsSize

- **Default Value** – 128
- **Data Type** – Integer

This option enables optimization to rewrite and combine compatible INSERT statements into batches. This value must increase exponentially by the power of 2.

This parameter is optional.

roleArn

- **Default Value** – None
- **Data Type** – String

The Amazon Resource Name (ARN) of role. Make sure to specify this parameter when you specify BasicJwtCredentialsProvider for the Plugin_Name option. You specify the ARN in the following format:

`arn:`*`partition`*`:`*`service`*`:`*`region`*`:`*`account-id`*`:`*`resource-id`*

This parameter is required if you specify BasicJwtCredentialsProvider for the Plugin_Name option.

roleSessionName

- **Default Value** – jwt_redshift_session
- **Data Type** – String

An identifier for the assumed role session. Typically, you pass the name or identifier that is associated with the user of your application. The temporary security credentials that your application uses are associated with that user. You can specify this parameter when you specify BasicJwtCredentialsProvider for the Plugin_Name option.

This parameter is optional.

scope

- **Default Value** – None
- **Data Type** – String

A space-separated list of scopes to which the user can consent. You specify this parameter so that your Microsoft Azure application can get consent for APIs that you want to call. You can specify this parameter when you specify BrowserAzureOAuth2CredentialsProvider for the Plugin_Name option.

This parameter is required for the BrowserAzureOAuth2CredentialsProvider plug-in.

SecretAccessKey

- **Default Value** – None
- **Data Type** – String

The IAM access key for the user or role. If this is specified, then AccessKeyID must also be specified. If passed in the JDBC URL, SecretAccessKey must be URL encoded.

This parameter is optional.

SessionToken

- **Default Value** – None
- **Data Type** – String

The temporary IAM session token associated with the IAM role you are using to authenticate. If passed in the JDBC URL, the temporary IAM session token must be URL encoded.

This parameter is optional.

serverlessAcctId

- **Default Value** – null
- **Data Type** – String

The Amazon Redshift Serverless account ID. The driver attempts to detect this parameter from the given host. If you're using a Network Load Balancer (NLB), the driver will fail to detect it, so you can set it here.

This parameter is optional.

serverlessWorkGroup

- **Default Value** – null
- **Data Type** – String

The Amazon Redshift Serverless workgroup name. The driver attempts to detect this parameter from the given host. If you're using a Network Load Balancer (NLB), the driver will fail to detect it, so you can set it here.

This parameter is optional.

socketFactory

- **Default Value** – null
- **Data Type** – String

This option specifies a socket factory for socket creation.

This parameter is optional.

socketTimeout

- **Default Value** – 0

- **Data Type** – Integer

The number of seconds to wait during socket read operations before timing out. If the operation takes longer than this threshold, then the connection is closed. When this property is set to 0, the connection doesn't time out.

This parameter is optional.

SSL

- **Default Value** – TRUE
- **Data Type** – String

Use this property to turn on or turn off SSL for the connection.

This parameter is optional.

You can specify the following values:

TRUE

The driver connects to the server through SSL.

FALSE

The driver connects to the server without using SSL. This option is not supported with IAM authentication.

Alternatively, you can configure the AuthMech property.

SSL_Insecure

- **Default Value** – true
- **Data Type** – String

This property indicates whether the IDP hosts server certificate should be verified.

This parameter is optional.

You can specify the following values:

true

The driver doesn't check the authenticity of the IDP server certificate.

false

The driver checks the authenticity of the IDP server certificate.

SSLCert

- **Default Value** – None
- **Data Type** – String

The full path of a .pem or .crt file containing additional trusted CA certificates for verifying the Amazon Redshift server instance when using SSL.

This parameter is required if SSLKey is specified.

SSLFactory

- **Default Value** – None
- **Data Type** – String

The SSL factory to use when connecting to the server through TLS/SSL without using a server certificate.

SSLKey

- **Default Value** – None
- **Data Type** – String

The full path of the .der file containing the PKCS8 key file for verifying the certificates specified in SSLCert.

This parameter is required if SSLCert is specified.

SSLMode

- **Default Value** – verify-full

- **Data Type** – String

Use this property to specify how the driver validates certificates when TLS/SSL is enabled.

This parameter is optional.

You can specify the following values:

verify-ca

The driver verifies that the certificate comes from a trusted certificate authority (CA).

verify-full

The driver verifies that the certificate comes from a trusted CA and that the host name in the certificate matches the host name specified in the connection URL.

> **Note**
>
> We recommend using `verify-full` for SSL connections whenever possible.

In driver versions 2.3.0 and later, the default is `verify-full`. In earlier versions, the default was `verify-ca`. If you don't set SSLMode and the server certificate doesn't match the host name, the driver logs a warning and connects. If you set SSLMode, the driver enforces the mode that you set.

SSLPassword

- **Default Value** – 0
- **Data Type** – String

The password for the encrypted key file specified in SSLKey.

This parameter is required if SSLKey is specified and the key file is encrypted.

SSLRootCert

- **Default Value** – None
- **Data Type** – String

The full path of a .pem or .crt file containing the root CA certificate for verifying the Amazon Redshift Server instance when using SSL.

StsEndpointUrl

- **Default Value** – null
- **Data Type** – String

You can specify an AWS Security Token Service (AWS STS) endpoint. If you specify this option, the Region option is ignored. You can only specify a secure protocol (HTTPS) for this endpoint.

tcpKeepAlive

- **Default Value** – TRUE
- **Data Type** – String

Use this property to turn on or turn off TCP keepalives.

This parameter is optional.

You can specify the following values:

TRUE

The driver uses TCP keepalives to prevent connections from timing out.

FALSE

The driver doesn't use TCP keepalives.

token

- **Default Value** – None
- **Data Type** – String

An AWS IAM Identity Center provided access token or an OpenID Connect (OIDC) JSON Web Token (JWT) provided by a web identity provider that's linked with AWS IAM Identity Center. Your application must generate this token by authenticating the user of your application with AWS IAM Identity Center or an identity provider linked with AWS IAM Identity Center.

This parameter works with `IdpTokenAuthPlugin`.

token_type

- **Default Value** – None
- **Data Type** – String

The type of token that is being used in `IdpTokenAuthPlugin`.

You can specify the following values:

ACCESS_TOKEN

Enter this if you use an AWS IAM Identity Center provided access token.

EXT_JWT

Enter this if you use an OpenID Connect (OIDC) JSON Web Token (JWT) provided by a web-based identity provider that's integrated with AWS IAM Identity Center.

This parameter works with `IdpTokenAuthPlugin`.

UID

- **Default Value** – None
- **Data Type** – String

The database username that you use to access the database.

This parameter is required.

User

- **Default Value** – None
- **Data Type** – String

When connecting using IAM authentication through an IDP, this is the username for the idp_host server. When using standard authentication, this can be used for the Amazon Redshift database username.

This parameter is optional.

webIdentityToken

- **Default Value** – None
- **Data Type** – String

The OAuth 2.1 access token or OpenID Connect ID token that is provided by the identity provider. Your application must get this token by authenticating the user of your application with a web identity provider. Make sure to specify this parameter when you specify BasicJwtCredentialsProvider for the Plugin_Name option.

This parameter is required if you specify BasicJwtCredentialsProvider for the Plugin_Name option.

Previous versions of JDBC driver version 2.x

Download a previous version of the Amazon Redshift JDBC driver version 2.x only if your tool requires a specific version of the driver.

These are the previous JDBC 4.2–compatible JDBC driver version 2.x drivers:

- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.2.9/redshift-jdbc42-2.2.9.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.2.8/redshift-jdbc42-2.2.8.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.2.7/redshift-jdbc42-2.2.7.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.2.6/redshift-jdbc42-2.2.6.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.2.5/redshift-jdbc42-2.2.5.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.2.4/redshift-jdbc42-2.2.4.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.2.3/redshift-jdbc42-2.2.3.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.2.2/redshift-jdbc42-2.2.2.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.2.1/redshift-jdbc42-2.2.1.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.2.0/redshift-jdbc42-2.2.0.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.34/redshift-jdbc42-2.1.0.34.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.33/redshift-jdbc42-2.1.0.33.zip

- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.32/redshift-jdbc42-2.1.0.32.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.30/redshift-jdbc42-2.1.0.30.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.29/redshift-jdbc42-2.1.0.29.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.28/redshift-jdbc42-2.1.0.28.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.26/redshift-jdbc42-2.1.0.26.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.25/redshift-jdbc42-2.1.0.25.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.24/redshift-jdbc42-2.1.0.24.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.23/redshift-jdbc42-2.1.0.23.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.22/redshift-jdbc42-2.1.0.22.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.21/redshift-jdbc42-2.1.0.21.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.20/redshift-jdbc42-2.1.0.20.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.19/redshift-jdbc42-2.1.0.19.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.18/redshift-jdbc42-2.1.0.18.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.17/redshift-jdbc42-2.1.0.17.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.16/redshift-jdbc42-2.1.0.16.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.15/redshift-jdbc42-2.1.0.15.zip

- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.14/redshift-jdbc42-2.1.0.14.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.13/redshift-jdbc42-2.1.0.13.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.12/redshift-jdbc42-2.1.0.12.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.11/redshift-jdbc42-2.1.0.11.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.10/redshift-jdbc42-2.1.0.10.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.9/redshift-jdbc42-2.1.0.9.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.8/redshift-jdbc42-2.1.0.8.zip
- https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.7/redshift-jdbc42-2.1.0.7.zip

Amazon Redshift Python connector

By using the Amazon Redshift connector for Python, you can integrate work with the AWS SDK for Python (Boto3), and also pandas and Numerical Python (NumPy). For more information on pandas, see the pandas GitHub repository. For more information on NumPy, see the NumPy GitHub repository.

The Amazon Redshift Python connector provides an open source solution. You can browse the source code, request enhancements, report issues, and provide contributions.

To use the Amazon Redshift Python connector, make sure that you have Python version 3.6 or later. For more information, see the Amazon Redshift Python driver license agreement.

The Amazon Redshift Python connector provides the following:

- AWS Identity and Access Management (IAM) authentication. For more information, see Identity and access management in Amazon Redshift.
- Identity provider authentication using federated API access. Federated API access is supported for corporate identity providers such as the following:
  - Azure AD. For more information, see the AWS Big Data blog post Federate Amazon Redshift access with Microsoft Azure AD single sign-on.

  - Active Directory Federation Services. For more information, see the AWS Big Data blog post Federate access to your Amazon Redshift cluster with Active Directory Federation Services (AD FS): Part 1.
  - Okta. For more information, see the AWS Big Data blog post Federate Amazon Redshift access with Okta as an identity provider.
  - PingFederate. For more information, see the PingFederate site.
  - JumpCloud. For more information, see the JumpCloud site.
- Amazon Redshift data types.

The Amazon Redshift Python connector implements Python Database API Specification 2.0. For more information, see PEP 249—Python Database API Specification v2.0 on the Python website.

Topics

- Installing the Amazon Redshift Python connector
- Configuration options for the Amazon Redshift Python connector
- Importing the Python connector
- Integrating the Python connector with NumPy
- Integrating the Python connector with pandas
- Using identity provider plugins
- Examples of using the Amazon Redshift Python connector
- API reference for the Amazon Redshift Python connector

Installing the Amazon Redshift Python connector

You can use any of the following methods to install the Amazon Redshift Python connector:

- Python Package Index (PyPI)
- Conda
- Cloning the GitHub repository

Installing the Python connector from PyPI

To install the Python connector from the Python Package Index (PyPI), you can use pip. To do this, run the following command.

```
>>> pip install redshift_connector
```

You can install the connector within a virtual environment. To do this, run the following command.

```
>>> pip install redshift_connector
```

Optionally, you can install pandas and NumPy with the connector.

```
>>> pip install 'redshift_connector[full]'
```

For more information on pip, see the pip site.

Installing the Python connector from Conda

You can install the Python connector from Anaconda.org.

```
>>>conda install -c conda-forge redshift_connector
```

Installing the Python connector by cloning the GitHub repository from AWS

To install the Python connector from source, clone the GitHub repository from AWS. After you install Python and virtualenv, set up your environment and install the required dependencies by running the following commands.

```
$ git clone https://github.com/aws/amazon-redshift-python-driver.git
$ cd amazon-redshift-python-driver
$ virtualenv venv
$ . venv/bin/activate
$ python -m pip install -r requirements.txt
$ python -m pip install -e .
$ python -m pip install redshift_connector
```

Configuration options for the Amazon Redshift Python connector

Following, you can find descriptions for the options that you can specify for the Amazon Redshift Python connector. The options below apply to the latest available connector version unless specified otherwise.

access_key_id

- **Default value** – None

- **Data type** – String

The access key for the IAM role or user configured for IAM database authentication.

This parameter is optional.

allow_db_user_override

- **Default value** – False
- **Data type** – Boolean

True

Specifies that the connector uses the `DbUser` value from the Security Assertion Markup Language (SAML) assertion.

False

Specifies that the value in the `DbUser` connection parameter is used.

This parameter is optional.

app_name

- **Default value** – None
- **Data type** – String

The name of the identity provider (IdP) application used for authentication.

This parameter is optional.

application_name

- **Default value** – None
- **Data type** – String

The name of the client application to pass to Amazon Redshift for audit purposes. The application name that you provide appears in the 'application_name' column of the SYS_CONNECTION_LOG table. This helps track and troubleshoot connection sources when debugging issues.

This parameter is optional.

auth_profile

- **Default value** – None
- **Data type** – String

The name of an Amazon Redshift authentication profile having connection properties as JSON. For more information about naming connection parameters, see the `RedshiftProperty` class. The `RedshiftProperty` class stores connection parameters provided by the end user and, if applicable, generated during the IAM authentication process (for example, temporary IAM credentials). For more information, see the RedshiftProperty class.

This parameter is optional.

auto_create

- **Default value** – False
- **Data type** – Boolean

A value that indicates whether to create the user if the user doesn't exist.

This parameter is optional.

client_id

- **Default value** – None
- **Data type** – String

The client ID from Azure IdP.

This parameter is optional.

client_secret

- **Default value** – None
- **Data type** – String

The client secret from Azure IdP.

This parameter is optional.

cluster_identifier

- **Default value** – None
- **Data type** – String

The cluster identifier of the Amazon Redshift cluster.

This parameter is optional.

credentials_provider

- **Default value** – None
- **Data type** – String

The IdP that is used for authenticating with Amazon Redshift. Following are valid values:

- `AdfsCredentialsProvider`
- `AzureCredentialsProvider`
- `BrowserAzureCredentialsProvider`
- `BrowserAzureOAuth2CredentialsProvider`
- `BrowserIdcAuthPlugin` – An authorization plugin using AWS IAM Identity Center.
- `BrowserSamlCredentialsProvider`
- `IdpTokenAuthPlugin` – An authorization plugin that accepts an AWS IAM Identity Center token or OpenID Connect (OIDC) JSON-based identity tokens (JWT) from any web identity provider linked to the AWS IAM Identity Center.
- `PingCredentialsProvider`
- `OktaCredentialsProvider`

This parameter is optional.

database

- **Default value** – None

- **Data type** – String

The name of the database to which you want to connect.

This parameter is required.

database_metadata_current_db_only

- **Default value** – True
- **Data type** – Boolean

A value that indicates whether an application supports multidatabase datashare catalogs. The default value of True indicates that the application doesn't support multidatabase datashare catalogs for backward compatibility.

This parameter is optional.

db_groups

- **Default value** – None
- **Data type** – String

A comma-separated list of existing database group names that the user indicated by DbUser joins for the current session.

This parameter is optional.

db_user

- **Default value** – None
- **Data type** – String

The user ID to use with Amazon Redshift.

This parameter is optional.

enable_table_types

- **Default value** – True

- **Data type** – Boolean

Specifies whether the connector recognizes detailed table type information from the data source in the results of the `get_tables` and `get_table_types` metadata methods. By default, the connector recognizes detailed table types.

- True: The connector recognizes the following table types: TABLE, VIEW, SYSTEM TABLE, SYSTEM VIEW, EXTERNAL TABLE, and LOCAL TEMPORARY.
- False: The connector normalizes the detailed table type information into the generic TABLE and VIEW table types.

This parameter is optional. It is available in connector versions 2.1.16 and later.

endpoint_url

- **Default value** – None
- **Data type** – String

The Amazon Redshift endpoint URL. This option is only for AWS internal use.

This parameter is optional.

group_federation

- **Default value** – False
- **Data type** – Boolean

This option specifies whether to use Amazon Redshift IDP groups.

This parameter is optional.

true

Use Amazon Redshift Identity Provider (IDP) groups.

false

Use STS API and GetClusterCredentials for user federation and specify **db_groups** for the connection.

host

- **Default value** – None
- **Data type** – String

The hostname of Amazon Redshift cluster.

This parameter is optional.

iam

- **Default value** – False
- **Data type** – Boolean

IAM authentication is enabled.

This parameter is required.

iam_disable_cache

- **Default value** – False
- **Data type** – Boolean

This option specifies whether the IAM credentials are cached. By default, the IAM credentials are cached. This improves performance when requests to the API gateway are throttled.

This parameter is optional.

idc_client_display_name

- **Default Value** – Amazon Redshift Python connector
- **Data Type** – String

The display name to be used for the client that's using BrowserIdcAuthPlugin.

This parameter is optional.

idc_region

- **Default Value** – None

- **Data Type** – String

The AWS region where the AWS IAM Identity Center instance is located.

This parameter is required only when authenticating using `BrowserIdcAuthPlugin` in the credentials_provider configuration option.

idp_partition

- **Default Value** – None
- **Data Type** – String

Specifies the cloud partition where your identity provider (IdP) is configured. This determines which IdP authentication endpoint the driver connects to.

If this parameter is left blank, the driver defaults to the commercial partition. Possible values are:

- `us-gov`: Use this value if your IdP is configured in Azure Government. For example, Azure AD Government uses the endpoint `login.microsoftonline.us`.
- `cn`: Use this value if your IdP is configured in the China cloud partition. For example, Azure AD China uses the endpoint `login.chinacloudapi.cn`.

This parameter is optional.

idpPort

- **Default value** – 7890
- **Data type** – Integer

The listen port to which IdP sends the SAML assertion.

This parameter is required.

idp_response_timeout

- **Default value** – 120
- **Data type** – Integer

The timeout for retrieving SAML assertion from IdP.

This parameter is required.

idp_tenant

- **Default value** – None
- **Data type** – String

The IdP tenant.

This parameter is optional.

issuer_url

- **Default Value** – None
- **Data Type** – String

Points to the AWS IAM Identity Center server's instance endpoint.

This parameter is required only when authenticating using `BrowserIdcAuthPlugin` in the credentials_provider configuration option.

listen_port

- **Default value** – 7890
- **Data type** – Integer

The port that the driver uses to receive the SAML response from the identity provider or authorization code when using SAML, Azure AD, or AWS IAM Identity Center services through a browser plugin.

This parameter is optional.

login_url

- **Default value** – None
- **Data type** – String

The single sign-on Url for the IdP.

This parameter is optional.

max_prepared_statements

- **Default value** – 1000
- **Data type** – Integer

The maximum number of prepared statements that will be cached per connection. Setting this parameter to 0 disables the caching mechanism. Entering a negative number for this parameter sets it to the default value.

This parameter is optional.

numeric_to_float

- **Default value** – False
- **Data type** – Boolean

This option specifies if the connector converts numeric data type values from decimal.Decimal to float. By default, the connector receives numeric data type values as decimal.Decimal and does not convert them.

We don't recommend enabling numeric_to_float for use cases that require precision, as results may be rounded.

For more information on decimal.Decimal and the tradeoffs between it and float, see decimal — Decimal fixed point and floating point arithmetic on the Python website.

This parameter is optional.

partner_sp_id

- **Default value** – None
- **Data type** – String

The Partner SP ID used for authentication with Ping.

This parameter is optional.

password

- **Default value** – None
- **Data type** – String

The password to use for authentication.

This parameter is optional.

port

- **Default value** – 5439
- **Data type** – Integer

The port number of the Amazon Redshift cluster.

This parameter is required.

preferred_role

- **Default value** – None
- **Data type** – String

The IAM role preferred for the current connection.

This parameter is optional.

principal_arn

- **Default value** – None
- **Data type** – String

The Amazon Resource Name (ARN) of the user or IAM role for which you are generating a policy. It's recommended that you attach a policy to a role and then assign the role to your user, for access.

This parameter is optional.

profile

- **Default value** – None

- **Data type** – String

The name of a profile in an AWS credentials file that contains AWS credentials.

This parameter is optional.

provider_name

- **Default value** – None
- **Data type** – String

The name of the Redshift Native Authentication Provider.

This parameter is optional.

region

- **Default value** – None
- **Data type** – String

The AWS Region where the cluster is located.

This parameter is optional.

role_arn

- **Default value** – None
- **Data type** – String

The Amazon Resource Name (ARN) of the role that the caller is assuming. This parameter is used by the provider indicated by `JwtCredentialsProvider`.

For the `JwtCredentialsProvider` provider, this parameter is mandatory. Otherwise, this parameter is optional.

role_session_name

- **Default value** – jwt_redshift_session
- **Data type** – String

An identifier for the assumed role session. Typically, you pass the name or identifier that is associated with the user who is using your application. The temporary security credentials that your application uses are associated with that user. This parameter is used by the provider indicated by `JwtCredentialsProvider`.

This parameter is optional.

scope

- **Default value** – None
- **Data type** – String

A space-separated list of scopes to which the user can consent. You specify this parameter so that your application can get consent for APIs that you want to call. You can specify this parameter when you specify BrowserAzureOAuth2CredentialsProvider for the credentials_provider option.

This parameter is required for the BrowserAzureOAuth2CredentialsProvider plug-in.

secret_access_key_id

- **Default value** – None
- **Data type** – String

The secret access key for the IAM role or user configured for IAM database authentication.

This parameter is optional.

session_token

- **Default value** – None
- **Data type** – String

The access key for the IAM role or user configured for IAM database authentication. This parameter is required if temporary AWS credentials are being used.

This parameter is optional.

serverless_acct_id

- **Default value** – None

- **Data type** – String

The Amazon Redshift Serverless account ID.

This parameter is optional.

serverless_work_group

- **Default value** – None
- **Data type** – String

The Amazon Redshift Serverless workgroup name.

This parameter is optional.

ssl

- **Default value** – True
- **Data type** – Boolean

Secure Sockets Layer (SSL) is enabled.

This parameter is required.

ssl_insecure

- **Default value** – False
- **Data type** – Boolean

A value that specifies whether to disable the verification of the IdP host's server SSL certificate. Setting this parameter to True will disable the verification of the IdP host's server SSL certificate. We recommend that you keep the default value of False in production environments.

This parameter is optional.

sslmode

- **Default value** – verify-full
- **Data type** – String

The security of the connection to Amazon Redshift. You can specify either of the following:

- verify-ca
- verify-full

> **Note**
>
> We recommend using `verify-full` for SSL connections whenever possible.

In driver versions 2.2.0 and later, the default is `verify-full`. In earlier versions, the default was `verify-ca`. If you don't set `sslmode` and the server certificate doesn't match the host name, the driver logs a warning and connects. If you set `sslmode`, the driver enforces the mode that you set.

This parameter is required.

tcp_keepalive

- **Default value** – True
- **Data type** – Boolean

Whether to use TCP keepalives to keep connections from timing out. You can specify the following values:

- True: The driver will use TCP keepalives to keep connections from timing out.
- False: The driver won't use TCP keepalives.

This parameter is optional.

tcp_keepalive_count

- **Default value** – None
- **Data type** – Integer

The number of unacknowledged probes to send before considering the connection inactive. For example, setting the value to 3 means that the driver will send 3 unanswered keepalive packets before determining that the connection is no longer active.

If this parameter is not specified, Amazon Redshift uses the system's default value.

This parameter is optional.

tcp_keepalive_interval

- **Default value** – None
- **Data type** – Integer

The interval, in seconds, between subsequent keepalive probes if the driver doesn't received acknowledgement for the probe before it. If you specify this parameter, it must be a positive integer.

If this parameter is not specified, Amazon Redshift uses the system's default value.

This parameter is optional.

tcp_keepalive_idle

- **Default value** – None
- **Data type** – Integer

The duration of inactivity, in seconds, after which the driver sends the first keepalive probe. For example, setting the value to 120 means that the driver will wait for 2 minutes of inactivity before sending the first keepalive packet. If you specify this parameter, it must be a positive integer.

If this parameter is not specified, Amazon Redshift uses the system's default value.

This parameter is optional.

timeout

- **Default value** – None
- **Data type** – Integer

The number of seconds before the connection to the server times out.

This parameter is optional.

token

- **Default Value** – None
- **Data Type** – String

An AWS IAM Identity Center provided access token or an OpenID Connect (OIDC) JSON Web Token (JWT) provided by a web identity provider that's linked with AWS IAM Identity Center. Your application must generate this token by authenticating the user of your application with AWS IAM Identity Center or an identity provider linked with AWS IAM Identity Center.

This parameter works with `IdpTokenAuthPlugin`.

token_type

- **Default Value** – None
- **Data Type** – String

The type of token that is being used in `IdpTokenAuthPlugin`.

You can specify the following values:

ACCESS_TOKEN

Enter this if you use an AWS IAM Identity Center provided access token.

EXT_JWT

Enter this if you use an OpenID Connect (OIDC) JSON Web Token (JWT) provided by a web-based identity provider that's integrated with AWS IAM Identity Center.

This parameter works with `IdpTokenAuthPlugin`.

user

- **Default value** – None
- **Data type** – String

The user name to use for authentication.

This parameter is optional.

web_identity_token

- **Default value** – None
- **Data type** – String

The OAuth 2.0 access token or OpenID Connect ID token that is provided by the identity provider. Make sure that your application gets this token by authenticating the user who is using your application with a web identity provider. The provider indicated by `JwtCredentialsProvider` uses this parameter.

For the `JwtCredentialsProvider` provider, this parameter is mandatory. Otherwise, this parameter is optional.

Importing the Python connector

To import the Python connector, run the following command.

```
>>> import redshift_connector
```

To connect to an Amazon Redshift cluster using AWS credentials, run the following command.

```
conn = redshift_connector.connect(
    host='examplecluster.abc123xyz789.us-west-1.redshift.amazonaws.com',
    port=5439,
    database='dev',
    user='awsuser',
    password='my_password'
 )
```

Integrating the Python connector with NumPy

Following is an example of integrating the Python connector with NumPy.

```
>>>  import numpy
#Connect to the cluster
>>> import redshift_connector
>>> conn = redshift_connector.connect(
     host='examplecluster.abc123xyz789.us-west-1.redshift.amazonaws.com',
     port=5439,
```

```
     database='dev',
     user='awsuser',
     password='my_password'
  )

# Create a Cursor object
>>> cursor = conn.cursor()

# Query and receive result set
cursor.execute("select * from book")

result: numpy.ndarray = cursor.fetch_numpy_array()
print(result)
```

Following is the result.

```
[['One Hundred Years of Solitude' 'Gabriel García Márquez']
['A Brief History of Time' 'Stephen Hawking']]
```

Integrating the Python connector with pandas

Following is an example of integrating the Python connector with pandas.

```
>>> import pandas

#Connect to the cluster
>>> import redshift_connector
>>> conn = redshift_connector.connect(
     host='examplecluster.abc123xyz789.us-west-1.redshift.amazonaws.com',
     port=5439,
     database='dev',
     user='awsuser',
     password='my_password'
  )

# Create a Cursor object
>>> cursor = conn.cursor()

# Query and receive result set
cursor.execute("select * from book")
result: pandas.DataFrame = cursor.fetch_dataframe()
print(result)
```

Using identity provider plugins

For general information on how to use identity provider plugins, see Options for providing IAM credentials. For more information about managing IAM identities, including best practices for IAM roles, see Identity and access management in Amazon Redshift.

Authentication using the ADFS identity provider plugin

Following is an example of using the Active Directory Federation Service (ADFS) identity provider plugin to authenticate a user connecting to an Amazon Redshift database.

```
>>> con = redshift_connector.connect(
    iam=True,
    database='dev',
    host='my-testing-cluster.abc.us-east-2.redshift.amazonaws.com',
    cluster_identifier='my-testing-cluster',
    credentials_provider='AdfsCredentialsProvider',
    user='brooke@myadfshostname.com',
    password='Hunter2',
    idp_host='myadfshostname.com'
)
```

Authentication using the Azure identity provider plugin

Following is an example of authentication using the Azure identity provider plugin. You can create values for a `client_id` and `client_secret` for an Azure Enterprise application as shown following.

```
>>>  con = redshift_connector.connect(
    iam=True,
    database='dev',
    host='my-testing-cluster.abc.us-east-2.redshift.amazonaws.com',
    cluster_identifier='my-testing-cluster',
    credentials_provider='AzureCredentialsProvider',
    user='brooke@myazure.org',
    password='Hunter2',
    idp_tenant='my_idp_tenant',
    client_id='my_client_id',
    client_secret='my_client_secret',
    preferred_role='arn:aws:iam:123:role/DataScientist'
)
```

Authentication using the AWS IAM Identity Center identity provider plugin

Following is an example of authentication using the AWS IAM Identity Center identity provider plugin.

```
with redshift_connector.connect(
credentials_provider='BrowserIdcAuthPlugin',
host='my-testing-cluster.abc.us-east-2.redshift.amazonaws.com',
database='dev',
idc_region='us-east-1',
issuer_url='https://identitycenter.amazonaws.com/ssoins-790723ebe09c86f9',
idp_response_timeout=60,
listen_port=8100,
idc_client_display_name='Test Display Name',
# port value of 5439 is specified by default
)
```

Authentication using Azure Browser identity provider plugin

Following is an example of using the Azure Browser identity provider plugin to authenticate a user connecting to an Amazon Redshift database.

Multi-factor authentication occurs in the browser, where the sign-in credentials are provided by the user.

```
>>>con = redshift_connector.connect(
    iam=True,
    database='dev',
    host='my-testing-cluster.abc.us-east-2.redshift.amazonaws.com',
    cluster_identifier='my-testing-cluster',
    credentials_provider='BrowserAzureCredentialsProvider',
    idp_tenant='my_idp_tenant',
    client_id='my_client_id',
)
```

Authentication using the Okta identity provider plugin

Following is an example of authentication using the Okta identity provider plugin. You can obtain the values for `idp_host`, `app_id` and `app_name` through the Okta application.

```
>>> con = redshift_connector.connect(
```

```
    iam=True,
    database='dev',
    host='my-testing-cluster.abc.us-east-2.redshift.amazonaws.com',
    cluster_identifier='my-testing-cluster',
    credentials_provider='OktaCredentialsProvider',
    user='brooke@myazure.org',
    password='hunter2',
    idp_host='my_idp_host',
    app_id='my_first_appetizer',
    app_name='dinner_party'
)
```

Authentication using JumpCloud with a generic SAML browser identity provider plugin

Following is an example of using JumpCloud with a generic SAML browser identity provider plugin for authentication.

The password parameter is required. However, you don't have to enter this parameter because multi-factor authentication occurs in the browser.

```
>>> con = redshift_connector.connect(
    iam=True,
    database='dev',
    host='my-testing-cluster.abc.us-east-2.redshift.amazonaws.com',
    cluster_identifier='my-testing-cluster',
    credentials_provider='BrowserSamlCredentialsProvider',
    user='brooke@myjumpcloud.org',
    password='',
    login_url='https://sso.jumpcloud.com/saml2/plustwo_melody'
)
```

Examples of using the Amazon Redshift Python connector

Following are examples of how to use the Amazon Redshift Python connector. To run them, you must first install the Python connector. For more information on installing the Amazon Redshift Python connector, see Installing the Amazon Redshift Python connector. For more information on configuration options you can use with the Python connector, see Configuration options for the Amazon Redshift Python connector.

Topics

- Connecting to and querying an Amazon Redshift cluster using AWS credentials

- Enabling autocommit
- Configuring cursor paramstyle
- Using COPY to copy data from an Amazon S3 bucket and UNLOAD to write data to it

Connecting to and querying an Amazon Redshift cluster using AWS credentials

The following example guides you through connecting to an Amazon Redshift cluster using your AWS credentials, then querying a table and retrieving the query results.

```
#Connect to the cluster
>>> import redshift_connector
>>> conn = redshift_connector.connect(
     host='examplecluster.abc123xyz789.us-west-1.redshift.amazonaws.com',
     database='dev',
     port=5439,
     user='awsuser',
     password='my_password'
  )

# Create a Cursor object
>>> cursor = conn.cursor()

# Query a table using the Cursor
>>> cursor.execute("select * from book")

#Retrieve the query result set
>>> result: tuple = cursor.fetchall()
>>> print(result)
 >> (['One Hundred Years of Solitude', 'Gabriel García Márquez'], ['A Brief History of
 Time', 'Stephen Hawking'])
```

Enabling autocommit

The autocommit property is off by default, following the Python Database API Specification. You can use the following commands to turn on the connection's autocommit property after performing a rollback command to make sure that a transaction is not in progress.

```
#Connect to the cluster
>>> import redshift_connector
>>> conn = redshift_connector.connect(...)
```

```
# Run a rollback command
>>>  conn.rollback()

# Turn on autocommit
>>>  conn.autocommit = True
>>>  conn.run("VACUUM")

# Turn off autocommit
>>>  conn.autocommit = False
```

Configuring cursor paramstyle

The paramstyle for a cursor can be modified via cursor.paramstyle. The default paramstyle used is `format`. Valid values for paramstyle are `qmark`, `numeric`, `named`, `format`, and `pyformat`.

The following are examples of using various paramstyles to pass parameters to a sample SQL statement.

```
# qmark
redshift_connector.paramstyle = 'qmark'
sql = 'insert into foo(bar, jar) VALUES(?, ?)'
cursor.execute(sql, (1, "hello world"))

# numeric
redshift_connector.paramstyle = 'numeric'
sql = 'insert into foo(bar, jar) VALUES(:1, :2)'
cursor.execute(sql, (1, "hello world"))

# named
redshift_connector.paramstyle = 'named'
sql = 'insert into foo(bar, jar) VALUES(:p1, :p2)'
cursor.execute(sql, {"p1":1, "p2":"hello world"})

# format
redshift_connector.paramstyle = 'format'
sql = 'insert into foo(bar, jar) VALUES(%s, %s)'
cursor.execute(sql, (1, "hello world"))

# pyformat
redshift_connector.paramstyle = 'pyformat'
sql = 'insert into foo(bar, jar) VALUES(%(bar)s, %(jar)s)'
cursor.execute(sql, {"bar": 1, "jar": "hello world"})
```

Using COPY to copy data from an Amazon S3 bucket and UNLOAD to write data to it

The following example shows how to copy data from an Amazon S3 bucket into a table and then unload from that table back into the bucket.

A text file named `category_csv.txt` containing the following data is uploaded to an Amazon S3 bucket:.

```
12,Shows,Musicals,Musical theatre
13,Shows,Plays,"All ""non-musical"" theatre"
14,Shows,Opera,"All opera, light, and ""rock"" opera"
15,Concerts,Classical,"All symphony, concerto, and choir concerts"
```

Following is an example of the Python code, which first connects to the Amazon Redshift database. It then creates a table called `category` and copies the CSV data from the S3 bucket into the table.

```
#Connect to the cluster and create a Cursor
>>> import redshift_connector
>>> with redshift_connector.connect(...) as conn:
>>> with conn.cursor() as cursor:

#Create an empty table
>>>     cursor.execute("create table category (catid int, cargroup varchar, catname
 varchar, catdesc varchar)")

#Use COPY to copy the contents of the S3 bucket into the empty table
>>>     cursor.execute("copy category from 's3://testing/category_csv.txt' iam_role
 'arn:aws:iam::123:role/RedshiftCopyUnload' csv;")

#Retrieve the contents of the table
>>>     cursor.execute("select * from category")
>>>     print(cursor.fetchall())

#Use UNLOAD to copy the contents of the table into the S3 bucket
>>>     cursor.execute("unload ('select * from category') to 's3://testing/
unloaded_category_csv.txt'  iam_role 'arn:aws:iam::123:role/RedshiftCopyUnload' csv;")

#Retrieve the contents of the bucket
>>>     print(cursor.fetchall())
 >> ([12, 'Shows', 'Musicals', 'Musical theatre'], [13, 'Shows', 'Plays', 'All "non-
musical" theatre'], [14, 'Shows', 'Opera', 'All opera, light, and "rock" opera'], [15,
 'Concerts', 'Classical', 'All symphony, concerto, and choir concerts'])
```

If you don't have `autocommit` set to true, commit with `conn.commit()` after running the `execute()` statements.

The data is unloaded into the file `unloaded_category_csv.text0000_part00` in the S3 bucket, with the following content:

```
12,Shows,Musicals,Musical theatre
13,Shows,Plays,"All ""non-musical"" theatre"
14,Shows,Opera,"All opera, light, and ""rock"" opera"
15,Concerts,Classical,"All symphony, concerto, and choir concerts"
```

API reference for the Amazon Redshift Python connector

Following, you can find a description of the Amazon Redshift Python connector API operations.

redshift_connector

Following, you can find a description of the `redshift_connector` API operation.

`connect(user, database, password[, port, …])`

Establishes a connection to an Amazon Redshift cluster. This function validates user input, optionally authenticates using an identity provider plugin, and then constructs a connection object.

`apilevel`

The DBAPI level supported, currently "2.0".

`paramstyle`, `str(object='') -> str str(bytes_or_buffer[, encoding[, errors]]) -> str`

The database API parameter style to use globally.

Connection

Following, you can find a description of the connection API operations for the Amazon Redshift Python connector.

`__init__(user, password, database[, host, …])`

Initializes a raw connection object.

`cursor`

Creates a cursor object bound to this connection.

`commit`

Commits the current database transaction.

`rollback`

Rolls back the current database transaction.

`close`

Closes the database connection.

`execute(cursor, operation, vals)`

Runs the specified SQL command. You can provide the parameters as a sequence or as a mapping, depending upon the value of `redshift_connector.paramstyle`.

`run(sql[, stream])`

Runs the specified SQL command. Optionally, you can provide a stream for use with the COPY command.

`xid(format_id, global_transaction_id, …)`

Create a transaction ID. Only the `global_transaction_id` parameter is used in postgres. format_id and branch_qualifier are not used in postgres. The `global_transaction_id` can be any string identifier supported by postgres that returns a tuple (`format_id`, `global_transaction_id`, `branch_qualifier`).

`tpc_begin(xid)`

Begins a TPC transaction with a transaction ID `xid` consisting of a a format ID, global transaction ID, and branch qualifier.

`tpc_prepare`

Performs the first phase of a transaction started with .tpc_begin.

`tpc_commit([xid])`

When called with no arguments, .tpc_commit commits a TPC transaction previously prepared with .tpc_prepare().

`tpc_rollback([xid])`

When called with no arguments, .tpc_rollback rolls back a TPC transaction.

`tpc_recover`

Returns a list of pending transaction IDs suitable for use with .tpc_commit(xid) or .tpc_rollback(xid).

Cursor

Following, you can find a description of the cursor API operation.

`__init__(connection[, paramstyle])`

Initializes a raw cursor object.

`insert_data_bulk(filename, table_name, parameter_indices, column_names, delimiter, batch_size)`

Runs a bulk INSERT statement.

`execute(operation[, args, stream, …])`

Runs a database operation.

`executemany(operation, param_sets)`

Prepares a database operation, and then runs it for all parameter sequences or mappings provided.

`fetchone`

Fetches the next row of a query result set.

`fetchmany([num])`

Fetches the next set of rows of a query result.

`fetchall`

Fetches all remaining rows of a query result.

`close`

Closes the cursor now.

`__iter__`

A cursor object can be iterated to retrieve the rows from a query.

`fetch_dataframe([num])`

Returns a dataframe of the last query results.

`write_dataframe(df, table)`

Writes the same structure dataframe into an Amazon Redshift database.

`fetch_numpy_array([num])`

Returns a NumPy array of the last query results.

`get_catalogs`

Amazon Redshift doesn't support multiple catalogs from a single connection. Amazon Redshift only returns the current catalog.

`get_tables([catalog, schema_pattern, …])`

Returns the unique public tables which are user-defined within the system.

`get_columns([catalog, schema_pattern, …])`

Returns a list of all columns in a specific table in an Amazon Redshift database.

AdfsCredentialsProvider plugin

Following is the syntax for the AdfsCredentialsProvider plugin API operation for the Amazon Redshift Python connector.

```
redshift_connector.plugin.AdfsCredentialsProvider()
```

AzureCredentialsProvider plugin

Following is the syntax for the AzureCredentialsProvider plugin API operation for the Amazon Redshift Python connector.

```
redshift_connector.plugin.AzureCredentialsProvider()
```

BrowserAzureCredentialsProvider plugin

Following is the syntax for the BrowserAzureCredentialsProvider plugin API operation for the Amazon Redshift Python connector.

```
redshift_connector.plugin.BrowserAzureCredentialsProvider()
```

BrowserSamlCredentialsProvider plugin

Following is the syntax for the BrowserSamlCredentialsProvider plugin API operation for the Amazon Redshift Python connector.

```
redshift_connector.plugin.BrowserSamlCredentialsProvider()
```

OktaCredentialsProvider plugin

Following is the syntax for the OktaCredentialsProvider plugin API operation for the Amazon Redshift Python connector.

```
redshift_connector.plugin.OktaCredentialsProvider()
```

PingCredentialsProvider plugin

Following is the syntax for the PingCredentialsProvider plugin API operation for the Amazon Redshift Python connector.

```
redshift_connector.plugin.PingCredentialsProvider()
```

SamlCredentialsProvider plugin

Following is the syntax for the SamlCredentialsProvider plugin API operation for the Amazon Redshift Python connector.

```
redshift_connector.plugin.SamlCredentialsProvider()
```

Amazon Redshift integration for Apache Spark

Apache Spark is a distributed processing framework and programming model that helps you do machine learning, stream processing, or graph analytics. Similar to Apache Hadoop, Spark is an open-source, distributed processing system commonly used for big data workloads. Spark has an optimized directed acyclic graph (DAG) execution engine and actively caches data in-memory. This can boost performance, especially for certain algorithms and interactive queries.

This integration provides you with a Spark connector you can use to build Apache Spark applications that read from and write to data in Amazon Redshift and Amazon Redshift Serverless. These applications don't compromise on application performance or transactional consistency of the data. This integration is automatically included in Amazon EMR and AWS Glue, so you can immediately run Apache Spark jobs that access and load data into Amazon Redshift as part of your data ingestion and transformation pipelines.

Currently, you can use the versions 3.3.x, 3.4.x, 3.5.x, 4.0.x, and 4.1.x of Spark with this integration.

This integration provides the following:

- AWS Identity and Access Management (IAM) authentication. For more information, see Identity and access management in Amazon Redshift.
- Predicate and query pushdown to improve performance.
- Amazon Redshift data types.
- Connectivity to Amazon Redshift and Amazon Redshift Serverless.

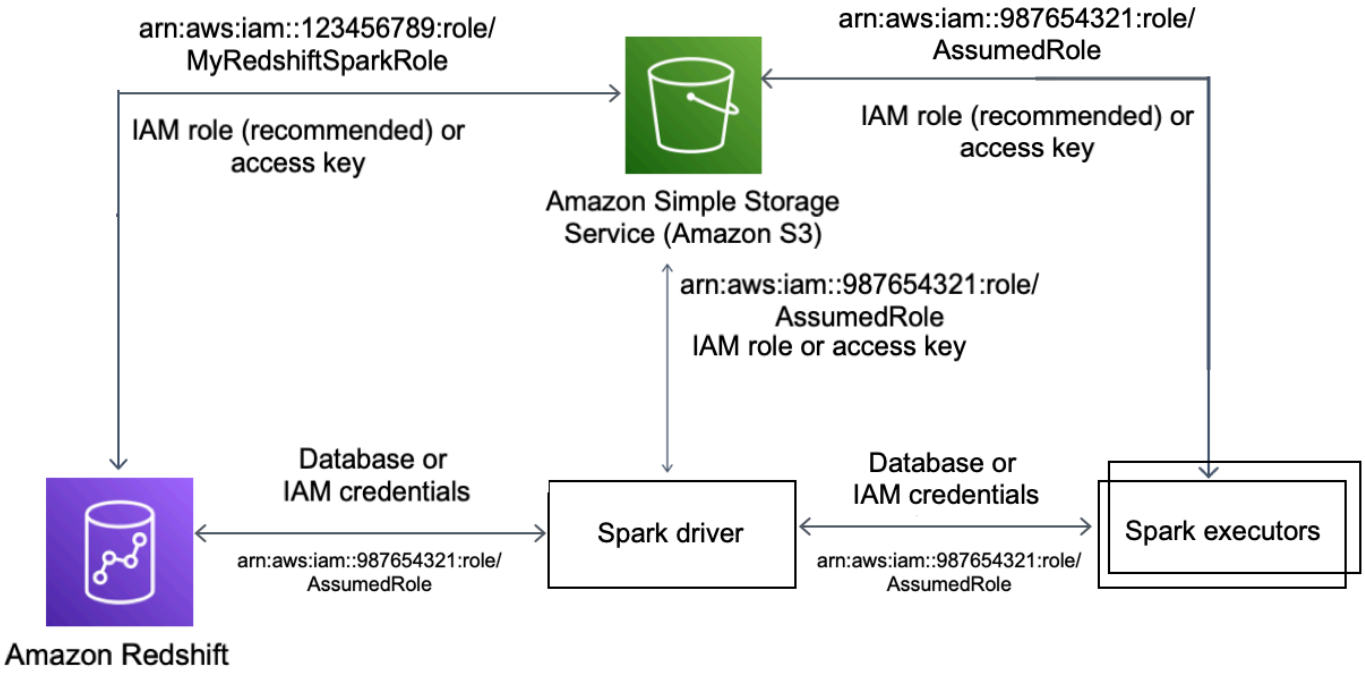
Considerations and limitations when using the Spark connector

- The tempdir URI points to an Amazon S3 location. This temp directory is not cleaned up automatically and could add additional cost. We recommend using Amazon S3 lifecycle policies in the *Amazon Simple Storage Service User Guide* to define the retention rules for the Amazon S3 bucket.
- By default, copies between Amazon S3 and Redshift don't work if the S3 bucket and Redshift cluster are in different AWS Regions. To use separate AWS Regions, set the `tempdir_region` parameter to the Region of the S3 bucket used for the `tempdir`.

- Cross-Region writes between S3 and Redshift if writing Parquet data using the `tempformat` parameter.
- We recommend using Amazon S3 server-side encryption to encrypt the Amazon S3 buckets used.
- We recommend blocking public access to Amazon S3 buckets.
- We recommend that the Amazon Redshift cluster should not be publicly accessible.
- We recommend turning on Amazon Redshift audit logging.
- We recommend turning on Amazon Redshift at-rest encryption.
- We recommend turning on SSL for the JDBC connection from Spark on Amazon EMR to Amazon Redshift.
- We recommend passing an IAM role using the parameter `aws_iam_role` for the Amazon Redshift authentication parameter.

Authentication with the Spark connector

The following diagram describes the authentication between Amazon S3, Amazon Redshift, the Spark driver, and Spark executors.



Authentication between Redshift and Spark

You can use the Amazon Redshift provided JDBC driver version 2.x driver to connect to Amazon Redshift with the Spark connector by specifying sign-in credentials. To use IAM, configure your JDBC url to use IAM authentication. To connect to a Redshift cluster from Amazon EMR or AWS Glue, make sure that your IAM role has the necessary permissions to retrieve temporary IAM credentials. The following list describes all of the permissions that your IAM role needs to retrieve credentials and run Amazon S3 operations.

- Redshift:GetClusterCredentials (for provisioned Redshift clusters)
- Redshift:DescribeClusters (for provisioned Redshift clusters)
- Redshift:GetWorkgroup (for Amazon Redshift Serverless workgroups)
- Redshift:GetCredentials (for Amazon Redshift Serverless workgroups)
- s3:ListBucket
- s3:GetBucket
- s3:GetObject
- s3:PutObject
- s3:GetBucketLifecycleConfiguration

For more information about GetClusterCredentials, see IAM policies for GetClusterCredentials.

You also must make sure that Amazon Redshift can assume the IAM role during COPY and UNLOAD operations.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "redshift.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
```

```
}
```

If you're using the latest JDBC driver, the driver will automatically manage the transition from an Amazon Redshift self-signed certificate to an ACM certificate. However, you must specify the SSL options to the JDBC url.

The following is an example of how to specify the JDBC driver URL and `aws_iam_role` to connect to Amazon Redshift.

```
df.write \
  .format("io.github.spark_redshift_community.spark.redshift ") \
  .option("url", "jdbc:redshift:iam://<the-rest-of-the-connection-string>") \
  .option("dbtable", "<your-table-name>") \
  .option("tempdir", "s3a://<your-bucket>/<your-directory-path>") \
  .option("aws_iam_role", "<your-aws-role-arn>") \
  .mode("error") \
  .save()
```

Authentication between Amazon S3 and Spark

If you're using an IAM role to authenticate between Spark and Amazon S3, use one of the following methods:

- The AWS SDK for Java will automatically attempt to find AWS credentials by using the default credential provider chain implemented by the DefaultAWSCredentialsProviderChain class. For more information, see Using the Default Credential Provider Chain.
- You can specify AWS keys via Hadoop configuration properties. For example, if your `tempdir` configuration points to a `s3n://` filesystem, set the `fs.s3n.awsAccessKeyId` and `fs.s3n.awsSecretAccessKey` properties in a Hadoop XML configuration file or call `sc.hadoopConfiguration.set()` to change Spark's global Hadoop configuration.

For example, if you are using the s3n filesystem, add:

```
sc.hadoopConfiguration.set("fs.s3n.awsAccessKeyId", "YOUR_KEY_ID")
sc.hadoopConfiguration.set("fs.s3n.awsSecretAccessKey", "YOUR_SECRET_ACCESS_KEY")
```

For the s3a filesystem, add:

```
sc.hadoopConfiguration.set("fs.s3a.access.key", "YOUR_KEY_ID")
```

```
sc.hadoopConfiguration.set("fs.s3a.secret.key", "YOUR_SECRET_ACCESS_KEY")
```

If you're using Python, use the following operations:

```
sc._jsc.hadoopConfiguration().set("fs.s3n.awsAccessKeyId", "YOUR_KEY_ID")
sc._jsc.hadoopConfiguration().set("fs.s3n.awsSecretAccessKey",
 "YOUR_SECRET_ACCESS_KEY")
```

- Encode authentication keys in the `tempdir` URL. For example, the URI `s3n://ACCESSKEY:SECRETKEY@bucket/path/to/temp/dir` encodes the key pair (`ACCESSKEY`, `SECRETKEY`).

Authentication between Redshift and Amazon S3

If you're using the COPY and UNLOAD commands in your query, you also must grant Amazon S3 access to Amazon Redshift to run queries on your behalf. To do so, first authorize Amazon Redshift to access other AWS services, then authorize the COPY and UNLOAD operations using IAM roles.

As a best practice, we recommend attaching permissions policies to an IAM role and then assigning it to users and groups as needed. For more information, see Identity and access management in Amazon Redshift.

Integration with AWS Secrets Manager

You can retrieve your Redshift username and password credentials from a stored secret in AWS Secrets Manager. To automatically supply Redshift credentials, use the `secret.id` parameter. For more information about how to create a Redshift credentials secret, see Create an AWS Secrets Manager database secret.

GroupID	ArtifactID	Supported Revision(s)	Description
com.amazonaws.secretsmanager	aws-secretsmanager-jdbc	1.0.12	The AWS Secrets Manager SQL Connection Library for Java lets Java Developers to easily connect to SQL databases using

GroupID	ArtifactID	Supported Revision(s)	Description
			secrets stored in AWS Secrets Manager.

Note

Acknowledgement: This documentation contains sample code and language developed by the Apache Software Foundation licensed under the Apache 2.0 license.

Performance improvements with pushdown

The Spark connector automatically applies predicate and query pushdown to optimize for performance. This support means that if you're using a supported function in your query, the Spark connector will turn the function into a SQL query and run the query in Amazon Redshift. This optimization results in less data being retrieved, so Apache Spark can process less data and have better performance. By default, pushdown is automatically activated. To deactivate it, set `autopushdown` to false.

```
import sqlContext.implicits._val
 sample= sqlContext.read
    .format("io.github.spark_redshift_community.spark.redshift")
    .option("url",jdbcURL )
    .option("tempdir", tempS3Dir)
    .option("dbtable", "event")
    .option("autopushdown", "false")
    .load()
```

The following functions are supported with pushdown. If you're using a function that's not in this list, the Spark connector will perform the function in Spark instead of Amazon Redshift, resulting in unoptimized performance. For a complete list of functions in Spark, see Built-in Functions.

- Aggregation functions
  - avg
  - count

- max
- min
- sum
- stddev_samp
- stddev_pop
- var_samp
- var_pop

- Boolean operators
  - in
  - isnull
  - isnotnull
  - contains
  - endswith
  - startswith
- Logical operators
  - and
  - or
  - not (or !)
- Mathematical functions
  - +
  - -
  - *
  - /
  - - (unary)
  - abs
  - acos
  - asin
  - atan
  - ceil
  - cos

  - exp
  - floor
  - greatest
  - least
  - log10
  - pi
  - pow
  - round
  - sin
  - sqrt
  - tan
- Miscellaneous functions
  - cast
  - coalesce
  - decimal
  - if
  - in
- Relational operators
  - !=
  - =
  - >
  - >=
  - <
  - <=
- String functions
  - ascii
  - lpad
  - rpad
  - translate
  - upper

  - lower
  - length
  - trim
  - ltrim
  - rtrim
  - like
  - substring
  - concat
- Time and date functions
  - add_months
  - date
  - date_add
  - date_sub
  - date_trunc
  - timestamp
  - trunc
- Mathematical operations
  - CheckOverflow
  - PromotePrecision
- Relational operations
  - Aliases (for example, AS)
  - CaseWhen
  - Distinct
  - InSet
  - Joins and cross joins
  - Limits
  - Unions, union all
  - ScalarSubquery
  - Sorts (ascending and descending)
  - UnscaledValue

Other configuration options

On this page, you can find descriptions for the options that you can specify for the Amazon Redshift Spark connector.

Maximum size of string columns

Redshift creates string columns as text columns when creating tables, which are stored as VARCHAR(256). If you want columns that support larger sizes, you can use maxlength to specify the maximum length of string columns. The following is an example of how to specify `maxlength`.

```
columnLengthMap.foreach { case (colName, length) =>
  val metadata = new MetadataBuilder().putLong("maxlength", length).build()
  df = df.withColumn(colName, df(colName).as(colName, metadata))
}
```

Column type

To set a column type, use the `redshift_type` field.

```
columnTypeMap.foreach { case (colName, colType) =>
  val metadata = new MetadataBuilder().putString("redshift_type", colType).build()
  df = df.withColumn(colName, df(colName).as(colName, metadata))
}
```

Compression encoding on a column

To use a specific compression encoding on a column, use the encoding field. For a full list of support compression encodings, see Compression encodings.

Description for a column

To set a description, use the `description` field.

Authentication between Redshift and Amazon S3

By default, the result is unloaded to Amazon S3 in the parquet format. To unload the result as pipe-delimited text file, specify the following option.

```
.option("unload_s3_format", "TEXT")
```

Pushdown statements

Parameter	Required	Default	Description
spark.datasource.redshift.community.autopushdown.lazyMode	No	True	Specifies whether the connector should lazily run pushdown statements Redshift. If true, the spark connector retrieves all of the related models and information before running the query, which generally yields better performance. If false, the spark connector runs pushdown statements immediately in the main Spark driver thread and is serialized across expressions.

Connector parameters

The parameter map or `OPTIONS` in Spark SQL supports the following settings.

Parameter	Required	Default	Description
dbtable	Yes, unless query is specified	N/A	The table to create or read from in Redshift.

Parameter	Required	Default	Description
			This parameter is required when saving data back to Redshift.
query	Yes, unless dbtable is specified	N/A	The query to read from in Redshift.
user	No	N/A	The Redshift username. Must be used with the password parameter. Valid only if the user and password are not parameters in the URL. Using both will cause an error.
password	No	N/A	The Redshift password. Must be used with the user parameter. Valid only if the user and password are not parameters in the URL. Using both will cause an error.

Parameter	Required	Default	Description
url	No	N/A	A JDBC URL. The format is jdbc:subprotocol://host:port/database?user=username&password=password. Subprotocol can be postgresql or Redshift, depending on which JDBC driver you have loaded. Note that one Redshift compatible driver must be in the classpath and match this URL. Host and port should point to the Redshift master node, so you must configure security groups and/or VPC to allow access from your driver application. Database is the Redshift database name. User and password are credentials to access the database, which must be embedded in this

Parameter	Required	Default	Description
			URL for JDBC, and the database user must have the necessary permissio ns to access the table.
aws_iam_role	Only if using IAM roles to authorize Redshift COPY/ UNLOAD operations	N/A	Fully specified ARN of the IAM Role attached to the Redshift cluster.
forward_spark_s3_c redentials	No	False	Indicates whether this library should automatically discover the credentials that Spark uses to connect to Amazon S3, and whether to forward those credentials to Redshift over the JDBC driver. These credentials are sent as part of the JDBC query. Therefore we recommend you enable SSL encryption with JDBC connection when using this option.

Parameter	Required	Default	Description
temporary_aws_access_key_id	No	N/A	AWS access key. Must have write permissions to the S3 bucket.
temporary_aws_secret_access_key	No	N/A	AWS secret access key corresponding to the access key.
temporary_aws_session_token	No	N/A	AWS session token corresponding to provided access key.
tempdir	No	N/A	A writeable location in Amazon S3. Used for unloading data when reading and Avro data to be loaded into Redshift when writing. If you're using a Redshift data source for Spark as part of a regular ETL pipeline, it can be useful to set a lifecycle policy on a bucket and use that as a temp location for this data.

Parameter	Required	Default	Description
jdbcdriver	No	Determined by the JDBC URL's subprotocol	The class name of the JDBC driver to use. This class must be on the classpath. In most cases, it should not be necessary to specify this option, as the appropriate driver classname should automatically be determined by the JDBC URL's subprotocol.
diststyle	No	Even	The Redshift Distribution Style to use when creating a table. Valid options are EVEN, KEY or ALL. When using KEY, you must also set a distribution key with the distkey option.
distkey	No, unless using DISTSTYLE_KEY	N/A	The name of a column in the table to use as the distribution key when creating a table.
sortkeyspec	No	N/A	A full Redshift Sort Key definition.

Parameter	Required	Default	Description
include_column_list	No	False	Indicates whether this library should automatically extract the columns from the schema and add them to the COPY command according to the Column mapping options.
description	No	N/A	A description for the table. The descripti on is set with the SQL COMMENT command, and appears in most query tools. See the `descripti on` metadata to set descriptions on individual columns.

Parameter	Required	Default	Description
preactions	No	N/A	A a semicolon-delimited list of SQL commands to be run before loading COPY command. It might be useful to run DELETE commands or similar before loading new data. If the command contains %s, the table name will be formatted in before runtime (in case you're using a staging table). If this command fails, it is treated as an exception. If you're using a staging table, the changes will be reverted and restore the backup table if preactions fail.

Parameter	Required	Default	Description
extracopyoptions	No	N/A	A list of extra options to append to the Redshift COPY command when loading data (such as `TRUNCATECOLUMNS` or `MAXERROR n`). See Optional parameter for a full list of available parameters. Note that since these options are appended to the end of the `COPY` command, you can only use options that make sense at the end of the command. That should cover most possible use cases.

Parameter	Required	Default	Description
sse_kms_key	No	N/A	The AWS KMS key ID to use for server-side encryption in S3 during the Redshift UNLOAD operation rather than the AWS default encryption. The Redshift IAM role must have access to the KMS key for writing with it, and the Spark IAM role must have access to the key for read operations. Reading the encrypted data requires no changes (AWS handles this) as long as Spark's IAM role has the proper access.
tempformat	No	AVRO	The format in which to save temporary files in Amazon S3 when writing to Redshift. Valid values are AVRO, CSV, CSV GZIP (compressed CSV), and PARQUET.

Parameter	Required	Default	Description
csvnullstring (experimental)	No	Null	The string value to write for nulls when using the CSV tempformat. This should be a value that does not appear in your actual data.
autopushdown	No	True	Indicates whether to apply predicate and query pushdown by capturing and analyzing the Spark logical plans for SQL operations. The operations are translated into a SQL query and then run in Redshift to improve performance.

Parameter	Required	Default	Description
autopushdown.s3_result_cache	No	False	Cache the query SQL to unload data Amazon S3 path mapping in memory, so that the same query doesn't need to run again in the same Spark session. Only supported when autopushdown is turned on. We do not recommend using this parameter when mixing read and write operations because cached results might contain stale information.
unload_s3_format	No	Parquet	The format with which to unload query results. Valid options are Parquet and Text, which specifies to unload query results in the pipe-delimited text format.

Parameter	Required	Default	Description
extraunloadoptions	No	N/A	Extra options to append to the Redshift UNLOAD command. Not all options are guaranteed to work as some options might conflict with other options set within the connector .
copydelay	No	30000	The delay (in ms) between retries for Redshift COPY operations.
copyretrycount	No	2	The number of times to retry Redshift COPY operations.

Parameter	Required	Default	Description
tempdir_region	No	N/A	The AWS region where `tempdir` is located. Setting this option improves connector performance for interactions with `tempdir` as well as automatically supply this value as part of the COPY and UNLOAD operations during the connector 's read and write operations. This setting is recommended in the following situations: 1) When the connector is running outside of AWS, as automatic Region discovery will fail and negatively affect connector performance. 2) When `tempdir` is in a different Region than the Redshift cluster, as using this setting alleviates the need

Parameter	Required	Default	Description
			to supply the Region manually using the `extracopy options` and `extraunlo adoptions` parameters. `tempdir` can't be in a different Region than the Redshift cluster when using PARQUET as the `tempformat` even if this using this parameter. 3) When the connector is running in a different Region than `tempdir`, as it improves the connector's access performance of `tempdir`.

Parameter	Required	Default	Description
secret.id	No	N/A	The name or ARN of your secret stored in AWS Secrets Manager. You can use this parameter to automatically supply Redshift credentials, but only if the user, password, and `DbUser` credentials are not passed into the JDBC URL or as other options.

Parameter	Required	Default	Description
secret.region	No	N/A	The primary AWS Region, such as US East (N. Virginia) , to search for the `secret.id` value. If you don't specify this Region, the connector will try to use the Default credential provider chain to resolve the Region of the `secret.id` . In some cases, such as if you're using the connector outside of an the connector will not be able to find the Region. We recommend using this setting in the folowing situations: 1) When the connector is running outside of AWS, as automatic Region discovery will fail and prevent authentication with Redshift When the connector is running in a

Parameter	Required	Default	Description
			different Region than `secret.id` , as it improves the connector's access performance of the secret.
secret.vpcEndpointUrl	No	N/A	The PrivateLink DNS endpoint URL for AWS Secrets Manager when overriding the Default credential provider chain.
secret.vpcEndpoint Region	No	N/A	The PrivateLink DNS endpoint Region for AWS Secrets Manager when overriding the Default credential provider chain.

Parameter	Required	Default	Description
jdbc.*	No	N/A	Additional parameters to pass to the underlying JDBC driver where the wildcard is the name of the JDBC parameter, such as jdbc.ssl. Note that the `jdbc` prefix will be removed before it is passed to the JDBC driver. To see all of the possible options for the Redshift JDBC driver, see Options for JDBC driver version 2.x configuration .

Parameter	Required	Default	Description
label	No	" "	An identifier to include in the query group set when running queries with the connector. Must be 100 or fewer characters, and all characters must be valid `unicodeId entifierParts` . If your identifier has more than 100 characters, the excess will be removed. When running a query with the connector, the query group will be set as a JSON format string, such as `{"spark-redshift-connector":{"svc":"","ver":"5.1.0-amzn-1-spark_3.3","op":"Read","lbl":""}}`) . This option replaces the value of the `lbl` key.

> **Note**
>
> Acknowledgement: This documentation contains sample code and language developed by the Apache Software Foundation licensed under the Apache 2.0 license.

Supported data types

The following data types in Amazon Redshift are supported with the Spark connector. For a complete list of supported data types in Amazon Redshift, see Data types. If a data type is not in the table below, it's not supported in the Spark connector.

Data type	Aliases
SMALLINT	INT2
INTEGER	INT, INT4
BIGINT	INT8
DECIMAL	NUMERIC
REAL	FLOAT4
DOUBLE PRECISION	FLOAT8, FLOAT
BOOLEAN	BOOL
CHAR	CHARACTER, NCHAR, BPCHAR
VARCHAR	CHARACTER VARYING, NVARCHAR, TEXT
DATE	
TIMESTAMP	Timestamp without time zone
TIMESTAMPTZ	Timestamp with time zone
SUPER	
TIME	Time without time zone

Data type	Aliases
TIMETZ	Time with time zone
VARBYTE	VARBINARY, BINARY VARYING

Complex data types

You can use the spark connector to read and write Spark complex data types such as `ArrayType`, `MapType`, and `StructType` to and from Redshift SUPER data type columns. If you provide a schema during a read operation, the data in the column will be converted to its corresponding complex types in Spark, including any nested types. Additionally, if `autopushdown` is enabled, projection of nested attributes, map values, and array indices are pushed down to Redshift so that the entire nested data structure no longer needs to be unloaded when accessing just a portion of the data.

When you write DataFrames from the connector, any column of type `MapType` (using `StringType`), `StructType`, or `ArrayType` is written to a Redshift SUPER data type column. When writing these nested data structures, the `tempformat` parameter must be of type `CSV`, `CSV GZIP`, or `PARQUET`. Using `AVRO` will cause an exception. Writing a `MapType` data structure that has a key type other than `StringType` will also cause an exception.

StructType

The following example demonstrates how to create a table with a SUPER data type that contains a struct

```
create table contains_super (a super);
```

You can then use the connector to query a `StringType` field `hello` from the SUPER column a in the table using a schema like in the following example.

```
import org.apache.spark.sql.types._

val sc = // existing SparkContext
val sqlContext = new SQLContext(sc)

val schema = StructType(StructField("a", StructType(StructField("hello",
 StringType) ::Nil)) :: Nil)
```

```
val helloDF = sqlContext.read
.format("io.github.spark_redshift_community.spark.redshift")
.option("url", jdbcURL )
.option("tempdir", tempS3Dir)
.option("dbtable", "contains_super")
.schema(schema)
.load().selectExpr("a.hello")
```

The following example demonstrates how to write a struct to the column `a`.

```
import org.apache.spark.sql.types._
import org.apache.spark.sql._

val sc = // existing SparkContext
val sqlContext = new SQLContext(sc)

val schema = StructType(StructField("a", StructType(StructField("hello",
 StringType) ::Nil)) :: Nil)
val data = sc.parallelize(Seq(Row(Row("world"))))
val mydf = sqlContext.createDataFrame(data, schema)

mydf.write.format("io.github.spark_redshift_community.spark.redshift").
option("url", jdbcUrl).
option("dbtable", tableName).
option("tempdir", tempS3Dir).
option("tempformat", "CSV").
mode(SaveMode.Append).save
```

MapType

If you prefer to use a `MapType` to represent your data, then you can use a `MapType` data structure in your schema and retrieve the value corresponding to a key in the map. Note that all keys in your `MapType` data structure must be of type String, and all of the values must of the same type, such as int.

The following example demonstrates how to get the value of the key `hello` in the column `a`.

```
import org.apache.spark.sql.types._

val sc = // existing SparkContext
val sqlContext = new SQLContext(sc)
```

```
val schema = StructType(StructField("a", MapType(StringType, IntegerType))::Nil)

val helloDF = sqlContext.read
    .format("io.github.spark_redshift_community.spark.redshift")
    .option("url", jdbcURL )
    .option("tempdir", tempS3Dir)
    .option("dbtable", "contains_super")
    .schema(schema)
    .load().selectExpr("a['hello']")
```

ArrayType

If the column contains an array instead of a struct, you can use the connector to query the first element in the array.

```
import org.apache.spark.sql.types._

val sc = // existing SparkContext
val sqlContext = new SQLContext(sc)

val schema = StructType(StructField("a", ArrayType(IntegerType)):: Nil)

val helloDF = sqlContext.read
    .format("io.github.spark_redshift_community.spark.redshift")
    .option("url", jdbcURL )
    .option("tempdir", tempS3Dir)
    .option("dbtable", "contains_super")
    .schema(schema)
    .load().selectExpr("a[0]")
```

Limitations

Using complex data types with the spark connector has the following limitations:

- All nested struct field names and map keys must be lowercase. If querying for complex field names with uppercase letters, you can try omitting the schema and using the `from_json` spark function to convert the returned string locally as a workaround.
- Any map fields used in read or write operations must have only `StringType` keys.
- Only `CSV`, `CSV GZIP`, and `PARQUET` are supported tempformat values for writing complex types to Redshift. Attempting to use `AVRO` will throw an exception.

Configuring a connection for ODBC driver version 2.x for Amazon Redshift

You can use an ODBC connection to connect to your Amazon Redshift cluster from many third-party SQL client tools and applications. If your client tool supports JDBC, you can choose to use that type of connection rather than ODBC due to the ease of configuration that JDBC provides. However, if your client tool doesn't support JDBC, you can follow the steps in this section to set up an ODBC connection on your client computer or Amazon EC2 instance.

Amazon Redshift provides 64-bit ODBC drivers for Linux, Windows and Mac operating systems; the 32-bit ODBC drivers are discontinued. Further updates to the 32-bit ODBC drivers will not be released, except for urgent security patches.

For the latest information about ODBC driver changes, see the change log.

Topics

- Getting the ODBC URL
- Using an Amazon Redshift ODBC driver on Microsoft Windows
- Using an Amazon Redshift ODBC driver on Linux
- Using an Amazon Redshift ODBC driver on Apple macOS
- Authentication methods
- Data types conversions
- ODBC driver options
- Previous ODBC driver versions
- Differences between the 2.x and 1.x versions of the ODBC driver

Getting the ODBC URL

Amazon Redshift displays the ODBC URL for your cluster in the Amazon Redshift console. This URL contains the information required to set up the connection between your client computer and the database.

An ODBC URL has the following format:

```
Driver={driver}; Server=endpoint_host; Database=database_name; UID=user_name;
 PWD=password; Port=port_number
```

The preceding format's fields have the following values:

Field	Value
Driver	The name of the 64-bit ODBC driver to use: **Amazon Redshift ODBC Driver (x64)**
Server	The endpoint host of the Amazon Redshift cluster.
Database	The database that you created for your cluster.
UID	The user name of a database user account that has permission to connect to the database. Although this value is a database-level permission and not a cluster-level permission, you can use the Redshift admin user account that you set up when you launched the cluster.
PWD	The password for the database user account to connect to the database.
Port	The port number that you specified when you launched the cluster. If you have a firewall, ensure that this port is open for you to use.

The following is an example ODBC URL:

```
Driver={Amazon Redshift ODBC Driver (x64)}; Server=examplecluster.abc123xyz789.us-
west-2.redshift.amazonaws.com; Database=dev; UID=adminuser;
 PWD=insert_your_admin_user_password_here; Port=5439
```

For information on where to find the ODBC URL, see Finding your cluster connection string.

Using an Amazon Redshift ODBC driver on Microsoft Windows

You must install the Amazon Redshift ODBC driver on client computers accessing an Amazon Redshift data warehouse. For each computer where you install the driver, there are the following minimum requirements:

- Administrator rights on the machine.
- The machine meets the following system requirements:

- One of the following operating systems:
  - Windows 10 or 8.1.
  - Windows Server 2025, 2022, 2019, 2016, or 2012.
- 100 MB of available disk space.
- Visual C++ Redistributable for Visual Studio 2015 for 64-bit Windows installed. You can download the installation package at Download Visual C++ Redistributable for Visual Studio 2022 on the Microsoft website.

Downloading and installing the Amazon Redshift ODBC driver

Use the following procedure to download and install the Amazon Redshift ODBC driver for Windows operating systems. Only use a different driver if you're running a third-party application that is certified for use with Amazon Redshift, and that application requires that specific driver.

To download and install the ODBC driver:

1. Download the following driver: 64-bit ODBC driver version 2.3.0.0

   The name for this driver is **Amazon Redshift ODBC Driver (x64)**.
2. Review the Amazon Redshift ODBC driver version 2.x license.
3. Double-click the .msi file, then follow the steps in the wizard to install the driver.

Creating a system DSN entry for an ODBC connection

After you download and install the ODBC driver, add a data source name (DSN) entry to the client computer or Amazon EC2 instance. SQL client tools can use this data source to connect to the Amazon Redshift database.

We recommend that you create a system DSN instead of a user DSN. Some applications load the data using a different database user account, and might not be able to detect user DSNs that are created under another database user account.

> **Note**
>
> For authentication using AWS Identity and Access Management (IAM) credentials or identity provider (IdP) credentials, additional steps are required. For more information, see Configure a JDBC or ODBC connection to use IAM credentials.

To create a system DSN entry for an ODBC connection:

1. In the **Start** menu, type "ODBC Data Sources." Choose **ODBC Data Sources**.

   Make sure that you choose the ODBC Data Source Administrator that has the same bitness as the client application that you are using to connect to Amazon Redshift.
2. In the **ODBC Data Source Administrator**, choose the **Driver** tab and locate the following driver folder: **Amazon Redshift ODBC Driver (x64)**.
3. Choose the **System DSN** tab to configure the driver for all users on the computer, or the **User DSN** tab to configure the driver for your database user account only.
4. Choose **Add**. The **Create New Data Source** window opens.
5. Choose the **Amazon Redshift ODBC driver (x64)**, and then choose **Finish**. The **Amazon Redshift ODBC Driver DSN Setup** window opens.
6. Under the **Connection Settings** section, enter the following information:
   - **Data source name**

     Enter a name for the data source. For example, if you followed the *Amazon Redshift Getting Started Guide*, you might type `exampleclusterdsn` to make it easy to remember the cluster that you associate with this DSN.
   - **Server**

     Specify the endpoint host for your Amazon Redshift cluster. You can find this information in the Amazon Redshift console on the cluster's details page. For more information, see Configuring connections in Amazon Redshift .
   - **Port**

     Enter the port number that the database uses. Depending on the port you selected when creating, modifying or migrating the cluster, allow access to the selected port.
   - **Database**

     Enter the name of the Amazon Redshift database. If you launched your cluster without specifying a database name, enter `dev`. Otherwise, use the name that you chose during the launch process. If you followed the *Amazon Redshift Getting Started Guide*, enter `dev`.
7. Under the **Authentication** section, specify the configuration options to configure standard or IAM authentication.
8. Choose **SSL Options** and specify a value for the following:

- **Authentication mode**

  Choose a mode for handling Secure Sockets Layer (SSL). In a test environment, you might use `prefer`. However, for production environments and when secure data exchange is required, use `verify-ca` or `verify-full`.

- **Min TLS**

  Optionally, choose the minimum version of TLS/SSL that the driver allows the data store to use for encrypting connections. For example, if you specify TLS 1.2, TLS 1.1 can't be used to encrypt connections. The default version is TLS 1.2.

9. In the **Proxy** tab, specify any proxy connection setting.
10. In the **Cursor** tab, specify options on how to return query results to your SQL client tool or application.
11. In **Advanced Options**, specify values for `logLevel`, `logPath`, `compression`, and other options.
12. Choose **Test**. If the client computer can connect to the Amazon Redshift database, the following message appears: **Connection successful**. If the client computer fails to connect to the database, you can troubleshoot possible issues by generating a log file and contacting AWS support. For information on generating logs, see (LINK).
13. Choose **OK**.

Using an Amazon Redshift ODBC driver on Linux

You must install the Amazon Redshift ODBC driver on client computers accessing an Amazon Redshift data warehouse. For each computer where you install the driver, there are the following minimum requirements:

- Root access on the machine.
- One of the following distributions:
  - Amazon Linux 2
  - Amazon Linux 2023
  - Red Hat® Enterprise Linux® (RHEL) 8 or later
  - CentOS 8 or later
  - Debian 11 or later
- 150 MB of available disk space.

- unixODBC 2.2.14 or later.
- glibc 2.26 or later.

Downloading and installing the Amazon Redshift ODBC driver

To download and install the Amazon Redshift ODBC driver version 2.x for Linux:

1. Download the following driver:
   - x86 64-bit RPM driver version 2.3.0.0
     - Linux artifact signature key: Key
     - Linux artifact signed hash: Hash
   - ARM 64-bit RPM driver version 2.3.0.0
     - Linux artifact signature key: Key
     - Linux artifact signed hash: Hash
   - x86 64-bit DEB driver version 2.3.0.0
     - Linux artifact signature key: Key
     - Linux artifact signed hash: Hash
   - ARM 64-bit DEB driver version 2.3.0.0
     - Linux artifact signature key: Key
     - Linux artifact signed hash: Hash

   > **Note**
   >
   > 32-bit ODBC drivers are discontinued. Further updates will not be released, except for urgent security patches.

2. Before installing, verify that AWS signed the driver package. After downloading the package, its signature key (certificate), and its signed hash (detached signature), run the following commands:

```
openssl x509 -in SignatureKeyFile -pubkey -noout > redshift-odbc-pubkey.pem
openssl dgst -sha256 -verify redshift-odbc-pubkey.pem \
    -signature SignedHashFile \
    PackageFile
```

If the package is authentic, `openssl` prints `Verified OK`. If the output is anything else, do not install the package. Re-download the files or contact AWS Support.

For example, to verify the x86 64-bit RPM:

```
openssl x509 -in AmazonRedshiftODBC-64-bit-2.3.0.0.x86_64-certificate.pem -pubkey -
noout > redshift-odbc-pubkey.pem
openssl dgst -sha256 -verify redshift-odbc-pubkey.pem \
    -signature AmazonRedshiftODBC-64-bit-2.3.0.0.x86_64-signature.bin \
    AmazonRedshiftODBC-64-bit-2.3.0.0.x86_64.rpm
```

3. Go to the location where you downloaded the package, and then run one of the following commands. Use the command that corresponds to your Linux distribution.

   On RHEL and CentOS operating systems (RPM), run the following command:

   ```
   sudo yum --nogpgcheck localinstall RPMFileName
   ```

   Replace `RPMFileName` with the RPM package file name. For example, the following command demonstrates installing the 64-bit driver:

   ```
   sudo yum --nogpgcheck localinstall AmazonRedshiftODBC-64-bit-2.x.xx.xxxx.x86_64.rpm
   ```

   On Debian and Ubuntu operating systems (DEB), navigate to the directory containing the downloaded .deb file and run the following command:

   ```
   sudo apt install ./DEBFileName
   ```

   Replace `DEBFileName` with the DEB package file name. For example:

   ```
   sudo apt install ./AmazonRedshiftODBC-64-bit-2.x.xx.xxxx.amd64.deb
   ```

Using an ODBC driver manager to configure the ODBC driver

On Linux, you use an ODBC driver manager to configure the ODBC connection settings. ODBC driver managers use configuration files to define and configure ODBC data sources and drivers. The ODBC driver manager that you use depends on the operating system that you use.

Configuring the ODBC driver using unixODBC driver manager

The following files are required to configure the Amazon Redshift ODBC driver:

- `amazon.redshiftodbc.ini`
- `odbc.ini`
- `odbcinst.ini`

If you installed to the default location, the `amazon.redshiftodbc.ini` configuration file is located in `/opt/amazon/redshiftodbcx64`.

Additionally, under `/opt/amazon/redshiftodbcx64`, you can find sample `odbc.ini` and `odbcinst.ini` files. You can use these files as examples for configuring the Amazon Redshift ODBC driver and the data source name (DSN).

We don't recommend using the Amazon Redshift ODBC driver installation directory for the configuration files. The sample files in the installed directory are for example purposes only. If you reinstall the Amazon Redshift ODBC driver at a later time, or upgrade to a newer version, the installation directory is overwritten. You will lose any changes that you might have made to files in the installation directory.

To avoid this, copy the `amazon.redshiftodbc.ini` file to a directory other than the installation directory. If you copy this file to the user's home directory, add a period (.) to the beginning of the file name to make it a hidden file.

For the `odbc.ini` and `odbcinst.ini` files, either use the configuration files in the user's home directory or create new versions in another directory. By default, your Linux operating system should have an `odbc.ini` file and an `odbcinst.ini` file in the user's home directory (`/home/$USER` or `~/.`). These default files are hidden files, which is indicated by the dot (.) in front of each file name. These files display only when you use the `-a` flag to list the directory contents.

Whichever option you choose for the `odbc.ini` and `odbcinst.ini` files, modify the files to add driver and DSN configuration information. If you create new files, you also need to set environment variables to specify where these configuration files are located.

By default, ODBC driver managers are configured to use hidden versions of the `odbc.ini` and `odbcinst.ini` configuration files (named `.odbc.ini` and `.odbcinst.ini`) located in the home directory. They also are configured to use the `amazon.redshiftodbc.ini` file in the driver installation directory. If you store these configuration files elsewhere, set the environment variables described following so that the driver manager can locate the files.

If you are using unixODBC, do the following:

- Set `ODBCINI` to the full path and file name of the `odbc.ini` file.
- Set `ODBCSYSINI` to the full path of the directory that contains the `odbcinst.ini` file.
- Set `AMAZONREDSHIFTODBCINI` to the full path and file name of the `amazon.redshiftodbc.ini` file.

The following is an example of setting the values above:

```
export ODBCINI=/usr/local/odbc/odbc.ini
export ODBCSYSINI=/usr/local/odbc
export AMAZONREDSHIFTODBCINI=/etc/amazon.redshiftodbc.ini
```

Configuring a connection using a data source name (DSN) on Linux

When connecting to your data store using a data source name (DSN), configure the `odbc.ini` file to define data source names (DSNs). Set the properties in the `odbc.ini` file to create a DSN that specifies the connection information for your data store.

On Linux operating systems, use the following format:

```
[ODBC Data Sources]
driver_name=dsn_name

[dsn_name]
Driver=path/driver_file
Host=cluster_endpoint
Port=port_number
Database=database_name
locale=locale
```

The following example shows the configuration for `odbc.ini` with the 64-bit ODBC driver on Linux operating systems.

```
[ODBC Data Sources]
Amazon_Redshift_x64=Amazon Redshift ODBC Driver (x64)

[Amazon_Redshift_x64]
Driver=/opt/amazon/redshiftodbcx64/librsodbc64.so
Host=examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com
Port=5932Database=dev
locale=en-US
```

Configuring a connection without a DSN on Linux

To connect to your data store through a connection that doesn't have a DSN, define the driver in the `odbcinst.ini` file. Then provide a DSN-less connection string in your application.

On Linux operating systems, use the following format:

```
[ODBC Drivers]
driver_name=Installed
...

[driver_name]
Description=driver_description
Driver=path/driver_file

...
```

The following example shows the configuration for `odbcinst.ini` with the 64-bit ODBC driver on Linux operating systems.

```
[ODBC Drivers]
Amazon Redshift ODBC Driver (x64)=Installed

[Amazon Redshift ODBC Driver (x64)]
Description=Amazon Redshift ODBC Driver (64-bit)
```

```
Driver=/opt/amazon/redshiftodbcx64/librsodbc64.so
```

Using an Amazon Redshift ODBC driver on Apple macOS

You must install the Amazon Redshift ODBC driver on client computers accessing an Amazon Redshift data warehouse. For each computer where you install the driver, these are the following minimum requirements:

- Root access on the machine.
- Apple macOS System Requirements:
  - A 64-bit version of Apple macOS version 11.7 or higher (such as Apple macOS Big Sur, Monterey, Ventura or later) is required. The Redshift ODBC driver only supports 64-bit client applications.
  - 150 MB of available disk space.
  - The driver supports applications built with iODBC 3.52.9+ or unixODBC 2.3.7+.

Downloading and installing the Amazon Redshift ODBC driver

Use the following procedure to download and install the Amazon Redshift ODBC driver on Apple macOS. Only use a different driver if you're running a third-party application that is certified for use with Amazon Redshift, and that application requires that specific driver.

To download and install the ODBC driver:

1. Download the following driver: 64-bit ODBC driver version 2.3.0.0

   This driver is supported on both x86_64 and arm64 architectures. The name for this driver is **Amazon Redshift ODBC Driver (x64)**.
2. Review the Amazon Redshift ODBC driver version 2.x license.
3. Double-click the .pkg file, then follow the steps in the wizard to install the driver. Alternatively, run the following command:

   ```
   sudo installer -pkg PKGFileName -target /
   ```

   Replace `PKGFileName` with the pkg package file name. For example, the following command demonstrates installing the 64-bit driver:

```
sudo installer -pkg ./AmazonRedshiftODBC-64-bit.X.X.XX.X.universal.pkg -target /
```

Using an ODBC driver manager to configure the ODBC driver

On Mac, you use an ODBC driver manager to configure the ODBC connection settings. ODBC driver managers use configuration files to define and configure ODBC data sources and drivers. The ODBC driver manager that you use depends on the operating system that you use.

Configuring the ODBC driver using iODBC or unixODBC driver manager

The following files are required to configure the Amazon Redshift ODBC driver:

- `amazon.redshiftodbc.ini`
- `odbc.ini`
- `odbcinst.ini`

If you installed to the default location, the `amazon.redshiftodbc.ini` configuration file is located in `/opt/amazon/redshiftodbcx64`.

Additionally, under `/opt/amazon/redshiftodbcx64`, you can find sample `odbc.ini` and `odbcinst.ini` files. You can use these files as examples for configuring the Amazon Redshift ODBC driver and the data source name (DSN). The sample files in the installed directory are for example purposes only.

We don't recommend using the Amazon Redshift ODBC driver installation directory for the configuration files. If you reinstall the Amazon Redshift ODBC driver at a later time, or upgrade to a newer version, the installation directory is overwritten. You will lose any changes that you might have made to files in the installation directory.

To avoid this, copy the `odbc.ini`, `odbcinst.ini` and `amazon.redshiftodbc.ini` files to a directory other than the installation directory. If you copy these files to the user's home directory, add a period (.) to the beginning of these file names to make it a hidden file.

Modify the files to add DSN configuration information. When you create new files, you also need to set environment variables to specify where these configuration files are located.

The following is an example of setting the environment variables:

```
export ODBCINI=/Library/ODBC/odbc.ini
export ODBCSYSINI=/Library/ODBC
export ODBCINSTINI=${ODBCSYSINI}/odbcinst.ini
```

For command-line applications: Add the export commands to your shell startup file (e.g., `~/.bash_profile` or `~/.zshrc`).

For supported version of driver manager, see here

Configuring a connection using a data source name (DSN) on Apple macOS

When connecting to your data store using a data source name (DSN), configure the `odbc.ini` file to define data source names (DSNs). Set the properties in the `odbc.ini` file to create a DSN that specifies the connection information for your Redshift data warehouse.

On Apple macOS, use the following format:

```
[ODBC Data Sources]
driver_name=dsn_name

[dsn_name]
Driver=path/driver_file
Host=cluster_endpoint
Port=port_number
Database=database_name
locale=locale
```

The following example shows the configuration for `odbc.ini` with the 64-bit ODBC driver on Apple macOS.

```
[ODBC Data Sources]
Amazon_Redshift_x64=Amazon Redshift ODBC Driver (x64)

[Amazon_Redshift_x64]
Driver=/opt/amazon/redshiftodbcx64/librsodbc64.dylib
Host=examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com
Port=5932
Database=dev
locale=en-US
```

Configuring a connection without a DSN on Apple macOS

To connect to your Redshift data warehouse through a connection that doesn't have a DSN, define the driver in the `odbcinst.ini` file. Then provide a DSN-less connection string in your application.

On Apple macOS, use the following format:

```
[ODBC Drivers]
driver_name=Installed
...

[driver_name]
Description=driver_description
Driver=path/driver_file

...
```

The following example shows the configuration for `odbcinst.ini` with the 64-bit ODBC driver on Apple macOS.

```
[ODBC Drivers]
Amazon Redshift ODBC Driver (x64)=Installed

[Amazon Redshift ODBC Driver (x64)]
Description=Amazon Redshift ODBC Driver (64-bit)
Driver=/opt/amazon/redshiftodbcx64/librsodbc64.dylib
```

Authentication methods

To protect data from unauthorized access, Amazon Redshift data stores require all connections to be authenticated using user credentials.

The following table illustrates the required and optional connection options for each authentication method that can be used to connect to the Amazon Redshift ODBC driver version 2.x:

Authentication Method	Required	Optional
Standard	• Host	

Authentication Method	Required	Optional
	• Port • Database • UID • Password	
IAM Profile	• Host • Port • Database • IAM • Profile	• ClusterID • Region • AutoCreate • EndpointURL • StsEndpointURL • InstanceProfile **Note** **ClusterID** and **Region** must be set in **Host** if they are not set separately.

Authentication Method	Required	Optional
IAM Credentials	• Host • Port • Database • IAM • AccessKeyID • SecretAccessKey	• ClusterID • Region • AutoCreate • EndpointURL • StsEndpointURL • SessionToken • UID **Note** **ClusterID** and **Region** must be set in **Host** if they are not set separately.
AD FS	• Host • Port • Database • IAM • plugin_name • UID • Password • IdP_Host • IdP_Port	• ClusterID • Region • AutoCreate • EndpointUrl • StsEndpointUrl • Preferred_Role • loginToRp • SSL_Insecure **Note** **ClusterID** and **Region** must be set in **Host** if they are not set separately.

Authentication Method	Required	Optional
Azure AD	• Host • Port • Database • IAM • plugin_name • UID • Password • IdP_Tenant • Client_ID • Client_Secret	• ClusterID • Region • AutoCreate • EndpointUrl • StsEndpointUrl • Preferred_Role • dbgroups_filter **Note** **ClusterID** and **Region** must be set in **Host** if they are not set separately.
JWT	• Host • Port • Database • IAM • plugin_name • web_identity_token	• provider_name

Authentication Method	Required	Optional
Okta	• Host • Port • Database • IAM • plugin_name • UID • Password • IdP_Host • App_Name • App_ID	• ClusterID • Region • AutoCreate • EndpointUrl • StsEndpointUrl • Preferred_Role **Note** **ClusterID** and **Region** must be set in **Host** if they are not set separately.
Ping Federate	• Host • Port • Database • IAM • plugin_name • UID • Password • IdP_Host • IdP_Port	• ClusterID • Region • AutoCreate • EndpointUrl • StsEndpointUrl • Preferred_Role • SSL_Insecure • partner_spid **Note** **ClusterID** and **Region** must be set in **Host** if they are not set separately.

Authentication Method	Required	Optional
Browser Azure AD	• Host • Port • Database • IAM • plugin_name • IdP_Tenant • Client_ID • UID	• ClusterID • Region • AutoCreate • EndpointUrl • StsEndpointUrl • Preferred_Role • dbgroups_filter • IdP_Response_Timeout • listen_port **Note** **ClusterID** and **Region** must be set in **Host** if they are not set separately.

Authentication Method	Required	Optional
Browser SAML	• Host • Port • Database • IAM • plugin_name • login_url • UID	• ClusterID • Region • AutoCreate • EndpointUrl • StsEndpointUrl • Preferred_Role • dbgroups_filter • IdP_Response_Timeout • listen_port **Note** **ClusterID** and **Region** must be set in **Host** if they are not set separately.
Auth Profile	• Host • Port • Database • AccessKeyID • SecretAccessKey	

Authentication Method	Required	Optional
Browser Azure AD OAUTH2	• Host • Port • Database • IAM • plugin_name • IdP_Tenant • Client_ID • UID	• ClusterID • Region • EndpointUrl • IdP_Response_Timeout • listen_port • scope • provider_name **Note** **ClusterID** and **Region** must be set in **Host** if they are not set separately.
AWS IAM Identity Center	• Host • Database • plugin_name • idc_region • issuer_url	• idc_client_display_name • idp_response_timeout • listen_port

Using an external credentials service

In addition to built-in support for AD FS, Azure AD, and Okta, the Windows version of the Amazon Redshift ODBC driver also provides support for other credentials services. The driver can authenticate connections using any SAML-based credential provider plugin of your choice.

To configure an external credentials service on Windows:

1. Create an IAM profile that specifies the credential provider plugin and other authentication parameters as needed. The profile must be ASCII-encoded, and must contain the following key-value pair, where `PluginPath` is the full path to the plugin application:

```
plugin_name = PluginPath
```

For example:

```
plugin_name = C:\Users\kjson\myapp\CredServiceApp.exe
```

For information on how to create a profile, see Using a Configuration Profile in the Amazon Redshift Cluster Management Guide.

2. Configure the driver to use this profile. The driver detects and uses the authentication settings specified in the profile.

Data types conversions

The Amazon Redshift ODBC driver version 2.x supports many common data formats, converting between Amazon Redshift and SQL data types.

The following table lists the supported data type mappings.

Amazon Redshift type	SQL type
BIGINT	SQL_BIGINT
BOOLEAN	SQL_BIT
CHAR	SQL_CHAR
DATE	SQL_TYPE_DATE
DECIMAL	SQL_NUMERIC
DOUBLE PRECISION	SQL_DOUBLE
GEOGRAPHY	SQL_ LONGVARBINARY
GEOMETRY	SQL_ LONGVARBINARY
INTEGER	SQL_INTEGER

Amazon Redshift type	SQL type
REAL	SQL_REAL
SMALLINT	SQL_SMALLINT
SUPER	SQL_LONGVARCHAR
TEXT	SQL_LONGVARCHAR
TIME	SQL_TYPE_TIME
TIMETZ	SQL_TYPE_TIME
TIMESTAMP	SQL_TYPE_ TIMESTAMP
TIMESTAMPTZ	SQL_TYPE_ TIMESTAMP
VARBYTE	SQL_LONGVARBINARY
VARCHAR	SQL_VARCHAR

ODBC driver options

You can use driver configuration options to control the behavior of the Amazon Redshift ODBC driver. Driver options are not case sensitive.

In Microsoft Windows, you typically set driver options when you configure a data source name (DSN). You can also set driver options in the connection string when you connect programmatically, or by adding or changing registry keys in `HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI\`*`your_DSN`*.

In Linux, you set driver configuration options in your `odbc.ini` and `amazon.redshiftodbc.ini` files. Configuration options set in an `amazon.redshiftodbc.ini` file apply to all connections. In contrast, configuration options set in an `odbc.ini` file are specific to a connection. Configuration options set in `odbc.ini` take precedence over configuration options set in `amazon.redshiftodbc.ini`.

Following are descriptions for the options that you can specify for the Amazon Redshift ODBC version 2.x driver:

Topics

- AccessKeyID
- app_id
- ApplicationName
- app_name
- AuthProfile
- AuthType
- AutoCreate
- BoolsAsChar
- CaFile
- client_id
- client_ secret
- ClusterId
- compression
- Database
- DatabaseMetadataCurrentDbOnly
- dbgroups_filter
- Driver
- DSN
- EnableTableTypes
- EndpointUrl
- Fetch
- ForceLowercase
- group_federation
- https_proxy_host
- https_proxy_password
- https_proxy_port
- https_proxy_username

- IAM
- idc_client_display_name
- idc_region
- idp_host
- idp_port
- idp_response_timeout
- idp_tenant
- idp_partition
- idp_use_https_proxy
- InstanceProfile
- issuer_url
- KeepAlive
- KeepAliveCount
- KeepAliveInterval
- KeepAliveIdle
- listen_port
- login_url
- loginToRp
- LogLevel
- LogPath
- LoginTimeout
- MaxLongVarcharSize
- MaxVarcharSize
- Min_TLS
- partner_spid
- Password | PWD
- plugin_name
- Port | PortNumber

- Prefer_PQ
- preferred_role
- Profile
- provider_name
- ProxyHost
- ProxyPort
- ProxyPwd
- ProxyUid
- ReadOnly
- region
- SecretAccessKey
- SessionToken
- Server | HostName | Host
- ssl_insecure
- SSLMode
- StreamingCursorRows
- StsConnectionTimeout
- StsEndpointUrl
- token
- token_type
- UID | User | LogonID
- UseDeclareFetch
- UseUnicode
- web_identity_token

AccessKeyID

- **Default Value** – None
- **Data Type** – String

The IAM access key for the user or role. If you set this parameter, you must also specify **SecretAccessKey**.

This parameter is optional.

app_id

- **Default Value** – None
- **Data Type** – String

The Okta-provided unique ID associated with your Amazon Redshift application.

This parameter is optional.

ApplicationName

- **Default value** – None
- **Data type** – String

The name of the client application to pass to Amazon Redshift for audit purposes. The application name that you provide appears in the 'application_name' column of the SYS_CONNECTION_LOG table. This helps track and troubleshoot connection sources when debugging issues.

This parameter is optional.

app_name

- **Default Value** – None
- **Data Type** – String

The name of the Okta application that you use to authenticate the connection to Amazon Redshift.

This parameter is optional.

AuthProfile

- **Default Value** – None
- **Data Type** – String

The authentication profile used to manage the connection settings. If you set this parameter, you must also set **AccessKeyID** and **SecretAccessKey**.

This parameter is optional.

AuthType

- **Default Value** – Standard
- **Data Type** – String

This option specifies the authentication mode that the driver uses when you configure a DSN using the Amazon Redshift ODBC Driver DSN Setup dialog box:

- Standard: Standard authentication using your Amazon Redshift user name and password.
- AWS Profile: IAM authentication using a profile.
- AWS IAM Credentials: IAM authentication using IAM credentials.
- Identity Provider: AD FS: IAM authentication using Active Directory Federation Services (AD FS).
- Identity Provider: Auth Plugin: An authorization plugin that accepts an AWS IAM Identity Center token or OpenID Connect (OIDC) JSON-based identity tokens (JWT) from any web identity provider linked to AWS IAM Identity Center.
- Identity Provider: Azure AD: IAM authentication using an Azure AD portal.
- Identity Provider: JWT: IAM authentication using a JSON Web Token (JWT).
- Identity Provider: Okta: IAM authentication using Okta.
- Identity Provider: PingFederate: IAM authentication using PingFederate.

This option is available only when you configure a DSN using the Amazon Redshift ODBC Driver DSN Setup dialog box in the Windows driver. When you configure a connection using a connection string or a non-Windows machine, the driver automatically determines whether to use Standard, AWS Profile, or AWS IAM Credentials authentication based on your specified credentials. To use an identity provider, you must set the **plugin_name** property.

This parameter is required.

AutoCreate

- **Default Value** – 0
- **Data Type** – Boolean

A boolean specifying whether the driver creates a new user when the specified user does not exist.

- 1 | TRUE: If the user specified by the **UID** does not exist, the driver creates a new user.
- 0 | FALSE: The driver does not create a new user. If the specified user does not exist, the authentication fails.

This parameter is optional.

BoolsAsChar

- **Default Value** – 0
- **Data Type** – Boolean

Specifies whether the driver reports BOOLEAN columns as character type instead of bit type.

- 1 | TRUE: The driver reports BOOLEAN columns as SQL_VARCHAR with a column size of 1. The driver returns Boolean values as the character strings "0" and "1".
- 0 | FALSE: The driver reports BOOLEAN columns as SQL_BIT. This is the default behavior.

Use this option if your application relies on BOOLEAN columns being returned as character data.

This parameter is optional. This option is available in driver versions 2.2.1 and later.

CaFile

- **Default Value** – None
- **Data Type** – String

The file path to the CA certificate file used for some forms of IAM authentication.

This parameter is only available on Linux.

This parameter is optional.

client_id

- **Default Value** – None
- **Data Type** – String

The client ID associated with your Amazon Redshift application in Azure AD.

This parameter is required if authenticating through the Azure AD service.

client_ secret

- **Default Value** – None
- **Data Type** – String

The secret key associated with your Amazon Redshift application in Azure AD.

This parameter is required if authenticating through the Azure AD service.

ClusterId

- **Default Value** – None
- **Data Type** – String

The name of the Amazon Redshift cluster you want to connect to. It is used in IAM authentication. The Cluster ID is not specified in the **Server** parameter.

This parameter is optional.

compression

- **Default Value** – off
- **Data Type** – String

The compression method used for wire protocol communication between the Amazon Redshift server and the client or driver.

You can specify the following values:

- lz4: Sets the compression method used for wire protocol communication with Amazon Redshift to `lz4`.
- zstd: Sets the compression method used for wire protocol communication with Amazon Redshift to `zstd`.
- off: Doesn't use compression for wire protocol communication with Amazon Redshift.

This parameter is optional.

Database

- **Default Value** – None
- **Data Type** – String

The name of the Amazon Redshift database that you want to access.

This parameter is required.

DatabaseMetadataCurrentDbOnly

- **Default Value** – 1
- **Data Type** – Boolean

A boolean specifying whether the driver returns metadata from multiple databases and clusters.

- 1 | TRUE: The driver only returns metadata from the current database.
- 0 | FALSE. The driver returns metadata across multiple Amazon Redshift databases and clusters.

This parameter is optional.

dbgroups_filter

- **Default Value** – None
- **Data Type** – String

The regular expression you can specify to filter out DbGroups that are received from the SAML response to Amazon Redshift when using Azure, Browser Azure, and Browser SAML authentication types.

This parameter is optional.

Driver

- **Default Value** – Amazon Redshift ODBC Driver (x64)
- **Data Type** – String

The name of the driver. The only supported value is **Amazon Redshift ODBC Driver (x64)**.

This parameter is required if you do not set **DSN**.

DSN

- **Default Value** – None
- **Data Type** – String

The name of the driver data source name. The application specifies the DSN in the SQLDriverConnect API.

This parameter is required if you do not set **Driver.**.

EnableTableTypes

- **Default Value** – 1
- **Data Type** – Boolean

Specifies whether the driver recognizes detailed table type information from the data source when an application calls the SQLTables catalog function. By default, the driver recognizes detailed table types.

- 1 | TRUE: The driver recognizes the following table types: TABLE, VIEW, SYSTEM TABLE, SYSTEM VIEW, EXTERNAL TABLE, and LOCAL TEMPORARY.
- 0 | FALSE: The driver normalizes the detailed table type information into the generic TABLE and VIEW table types.

This parameter is optional. It is available in driver versions 2.2.1 and later.

EndpointUrl

- **Default Value** – None
- **Data Type** – String

The overriding endpoint used to communicate with the Amazon Redshift Coral Service for IAM authentication.

This parameter is optional.

Fetch

- **Default Value** – 100
- **Data Type** – Integer

When Declare/Fetch mode is enabled, this option controls how many rows the driver retrieves per round-trip, letting you tune the balance between memory use and network overhead for your workload. A larger value fetches more rows at once, reducing the number of round-trips to the server at the cost of higher memory per fetch. A smaller value lowers memory use but increases how often the driver goes back to the server. It sets the number of rows the driver returns at a time when `UseDeclareFetch` is enabled. For more information, see UseDeclareFetch.

This parameter is optional.

ForceLowercase

- **Default Value** – 0
- **Data Type** – Boolean

A boolean specifying whether the driver lowercases all DbGroups sent from the identity provider to Amazon Redshift when using single sign-on authentication.

- 1 | TRUE: The driver lowercases all DbGroups that are sent from the identity provider.
- 0 | FALSE: The driver does not alter DbGroups.

This parameter is optional.

group_federation

- **Default Value** – 0
- **Data Type** – Boolean

A boolean specifying whether the `getClusterCredentialsWithIAM` API is used for obtaining temporary cluster credentials in provisioned clusters. This option lets IAM users integrate with Redshift database roles in provisioned clusters. Note that this option does not apply to Redshift Serverless namespaces.

- 1 | TRUE: The driver uses the `getClusterCredentialsWithIAM` API for obtaining temporary cluster credentials in provisioned clusters.
- 0 | FALSE: The driver uses the default `getClusterCredentials` API for obtaining temporary cluster credentials in provisioned clusters.

This parameter is optional.

https_proxy_host

- **Default Value** – None
- **Data Type** – String

The host name or IP address of the proxy server through which you want to pass IAM authentication processes.

This parameter is optional.

https_proxy_password

- **Default Value** – None
- **Data Type** – String

The password that you use to access the proxy server. It's used for IAM authentication.

This parameter is optional.

https_proxy_port

- **Default Value** – None
- **Data Type** – Integer

The number of the port that the proxy server uses to listen for client connections. It's used for IAM authentication.

This parameter is optional.

https_proxy_username

- **Default Value** – None

- **Data Type** – String

The user name that you use to access the proxy server. It's used for IAM authentication.

This parameter is optional.

IAM

- **Default Value** – 0
- **Data Type** – Boolean

A boolean specifying whether the driver uses an IAM authentication method to authenticate the connection.

- 1 | TRUE: The driver uses one of the IAM authentication methods (using an access key and secret key pair, or a profile, or a credentials service).
- 0 | FALSE. The driver uses standard authentication (using your database user name and password).

This parameter is optional.

idc_client_display_name

- **Default Value** – Amazon Redshift ODBC driver
- **Data Type** – String

The display name to be used for the client that's using BrowserIdcAuthPlugin.

This parameter is optional.

idc_region

- **Default Value** – None
- **Data Type** – String

The AWS region where the AWS IAM Identity Center instance is located.

This parameter is required only when authenticating using `BrowserIdcAuthPlugin` in the plugin_name configuration option.

idp_host

- **Default Value** – None
- **Data Type** – String

The IdP (identity provider) host you are using to authenticate into Amazon Redshift.

This parameter is optional.

idp_port

- **Default Value** – None
- **Data Type** – Integer

The port for an IdP (identity provider) you are using to authenticate into Amazon Redshift. Depending on the port you selected when creating, modifying or migrating the cluster, allow access to the selected port.

This parameter is optional.

idp_response_timeout

- **Default Value** – 120
- **Data Type** – Integer

The number of seconds that the driver waits for the SAML response from the identity provider when using SAML or Azure AD services through a browser plugin.

This parameter is optional.

idp_tenant

- **Default Value** – None
- **Data Type** – String

The Azure AD tenant ID associated with your Amazon Redshift application.

This parameter is required if authenticating through the Azure AD service.

idp_partition

- **Default Value** – None
- **Data Type** – String

Specifies the cloud partition where your identity provider (IdP) is configured. This determines which IdP authentication endpoint the driver connects to.

If this parameter is left blank, the driver defaults to the commercial partition. Possible values are:

- `us-gov`: Use this value if your IdP is configured in Azure Government. For example, Azure AD Government uses the endpoint `login.microsoftonline.us`.
- `cn`: Use this value if your IdP is configured in the China cloud partition. For example, Azure AD China uses the endpoint `login.chinacloudapi.cn`.

This parameter is optional.

idp_use_https_proxy

- **Default Value** – 0
- **Data Type** – Boolean

A boolean specifying whether the driver passes the authentication processes for identity providers (IdP) through a proxy server.

- 1 | TRUE: The driver passes IdP authentication processes through a proxy server.
- 0 | FALSE. The driver does not pass IdP authentication processes through a proxy server.

This parameter is optional.

InstanceProfile

- **Default Value** – 0
- **Data Type** – Boolean

A boolean specifying whether the driver uses the Amazon EC2 instance profile, when configured to use a profile for authentication.

- 1 | TRUE: The driver uses the Amazon EC2 instance profile.
- 0 | FALSE. The driver uses the chained roles profile specified by the Profile Name option (**Profile**) instead.

This parameter is optional.

issuer_url

- **Default Value** – None
- **Data Type** – String

Points to the AWS IAM Identity Center server's instance endpoint.

This parameter is required only when authenticating using `BrowserIdcAuthPlugin` in the plugin_name configuration option.

KeepAlive

- **Default Value** – 1
- **Data Type** – Boolean

A boolean specifying whether the driver uses TCP keepalives to prevent connections from timing out.

- 1 | TRUE: The driver uses TCP keepalives to prevent connections from timing out.
- 0 | FALSE. The driver does not use TCP keepalives.

This parameter is optional.

KeepAliveCount

- **Default Value** – 0
- **Data Type** – Integer

The number of TCP keepalive packets that can be lost before the connection is considered broken. When this parameter is set to 0, the driver uses the system default for this setting.

This parameter is optional.

KeepAliveInterval

- **Default Value** – 0
- **Data Type** – Integer

The number of seconds between each TCP keepalive retransmission. When this parameter is set to 0, the driver uses the system default for this setting.

This parameter is optional.

KeepAliveIdle

- **Default Value** – 0
- **Data Type** – Integer

The number of seconds of inactivity before the driver sends a TCP keepalive packet. When this parameter is set to 0, the driver uses the system default for this setting.

This parameter is optional.

listen_port

- **Default Value** – 7890
- **Data Type** – Integer

The port that the driver uses to receive the SAML response from the identity provider or authorization code when using SAML, Azure AD, or AWS IAM Identity Center services through a browser plugin.

This parameter is optional.

login_url

- **Default Value** – None

- **Data Type** – String

The URL for the resource on the identity provider's website when using the generic Browser SAML plugin.

This parameter is required if authenticating with the SAML or Azure AD services through a browser plugin.

loginToRp

- **Default Value** – urn:amazon:webservices
- **Data Type** – String

The relying party trust that you want to use for the AD FS authentication type.

This string is optional.

LogLevel

- **Default Value** – 0
- **Data Type** – Integer

> **Note**
>
> `LogLevel` and `LogPath` are global settings. The driver reads them in the following order of precedence:
>
> 1. **Connection string** (`LogLevel=6;LogPath=/path;`) — highest priority, overrides all other settings at connect time.
> 2. **On Windows:** The DSN Setup dialog (Advanced tab).
>
>   **On Linux and macOS:** `[ODBC]` section in your `odbc.ini` file.
> 3. **`[DRIVER]` section** in the `amazon.redshiftodbc.ini` file — used as a fallback if logging is not enabled from the above source.
>
> Setting `LogLevel` or `LogPath` in an individual DSN section (for example, `[MyDSN]`) has no effect. These options are not read from per-DSN sections.

Use this property to enable or disable logging in the driver and to specify the amount of detail included in log. files. We recommend you only enable logging long enough to capture an issue, as logging decreases performance and can consume a large quantity of disk space.

Set the property to one of the following values:

- 0: OFF. Disable all logging.
- 1: FATAL. Reserved for fatal errors.
- 2: ERROR. Logs error events such as connection failures and authentication errors.
- 3: WARN. Logs warnings and unexpected conditions in addition to errors.
- 4: INFO. Logs general progress information such as connection status and server parameters.
- 5: DEBUG. Logs all driver activity including ODBC API function calls with arguments and return values.
- 6: TRACE. Logs all driver activity including additional low-level protocol and internal state details.

> **Important**
>
> Log level 5 (DEBUG) or higher captures all driver activity, which may include sensitive information such as connection credentials and query data. Only enable these log levels temporarily to capture an issue, and ensure all sensitive information is redacted before sharing logs with support or any third party.

When you enable logging, the driver writes all driver activity to a `redshift_odbc.log` file in the location that you specify in the **LogPath** property.

Starting with driver version 2.3.0, the separate `redshift_odbc.log.1` connection handshake log is deprecated, and the driver no longer creates it.

This parameter is optional.

LogPath

- **Default Value** – The OS-specific TEMP directory
- **Data Type** – String

Note

`LogLevel` and `LogPath` are global settings. The driver reads them in the following order of precedence:

1. **Connection string** (`LogLevel=6;LogPath=/path;`) — highest priority, overrides all other settings at connect time.
2. **On Windows:** The DSN Setup dialog (Advanced tab).

   On Linux and macOS: `[ODBC]` section in your `odbc.ini` file.
3. **`[DRIVER]` section** in the `amazon.redshiftodbc.ini` file — used as a fallback if logging is not enabled from the above source.

Setting `LogLevel` or `LogPath` in an individual DSN section (for example, `[MyDSN]`) has no effect. These options are not read from per-DSN sections.

The full path to the folder where the driver saves log files when **LogLevel** is higher than 0.

This parameter is optional.

LoginTimeout

- **Default Value** – 0
- **Data Type** – Integer

The number of seconds the driver waits for a connection to the server to be established before timing out. A value of 0 means no timeout (the driver waits indefinitely).

You can set this option in the connection string (using `LoginTimeout` or the short alias `LT`), in the DSN configuration, or programmatically using `SQLSetConnectAttr` with `SQL_ATTR_LOGIN_TIMEOUT`. On Windows, this option is also available in the DSN Setup dialog (Advanced tab).

This parameter is optional.

MaxLongVarcharSize

- **Default Value** – 65535

- **Data Type** – Integer

The column size that the driver reports for columns reclassified as large text (SQL_LONGVARCHAR) by **MaxVarcharSize**. This value replaces the column's actual declared size in metadata responses.

The default of 65,535 is recommended because some middleware layers, such as MSDASQL, reject large-text columns whose reported size falls between 8,001 and 32,768 bytes. By reporting 65,535, the driver ensures that reclassified columns are accepted without errors. Most users do not need to change this value.

This parameter has no effect when **MaxVarcharSize** is set to 0 (reclassification disabled).

This parameter is optional. It is available in driver versions 2.1.16 and later.

MaxVarcharSize

- **Default Value** – 255
- **Data Type** – Integer

The maximum column size that the driver reports as a regular text column (SQL_VARCHAR). The driver reclassifies any CHAR and VARCHAR column larger than this value as large text (SQL_LONGVARCHAR or SQL_WLONGVARCHAR when **UseUnicode** is enabled).

This reclassification prevents errors in applications or middleware layers that cannot process regular-text columns beyond a certain size. For example, SQL Server linked servers that access Amazon Redshift through MSDASQL (OLE DB) cannot process SQL_VARCHAR columns larger than approximately 8,000 bytes, so reclassifying oversized columns as large text avoids conversion failures.

- 255 (default): Columns larger than 255 characters are reported as SQL_LONGVARCHAR.
- 0: Disables reclassification. All columns are reported as SQL_VARCHAR regardless of size.
- Custom value: Any positive integer. Columns larger than the specified value are reported as SQL_LONGVARCHAR.

When a column is reclassified, the driver reports its size using the value specified in **MaxLongVarcharSize**.

This parameter is optional. It is available in driver versions 2.1.16 and later.

Min_TLS

- **Default Value** – 1.2
- **Data Type** – String

The minimum version of TLS/SSL that the driver allows the data store to use for encrypting connections. For example, if TLS 1.2 is specified, TLS 1.1 cannot be used to encrypt connections.

Min_TLS accepts the following values:

- 1.0: The connection must use at least TLS 1.0.
- 1.1: The connection must use at least TLS 1.1.
- 1.2: The connection must use at least TLS 1.2.

This parameter is optional.

partner_spid

- **Default Value** – None
- **Data Type** – String

The partner SPID (service provider ID) value to use when authenticating the connection using the PingFederate service.

This parameter is optional.

Password | PWD

- **Default Value** – None
- **Data Type** – String

The password corresponding to the database user name that you provided in the User field (**UID | User | LogonID**).

This parameter is optional.

plugin_name

- **Default Value** – None

- **Data Type** – String

The credentials provider plugin name that you want to use for authentication.

The following values are supported:

- `ADFS`: Use Active Directory Federation Services for authentication.
- `AzureAD`: Use Microsoft Azure Active Directory (AD) Service for authentication.
- `BrowserAzureAD`: Use a browser plugin for the Microsoft Azure Active Directory (AD) Service for authentication.
- `BrowserIdcAuthPlugin` : An authorization plugin using AWS IAM Identity Center.
- `BrowserSAML`: Use a browser plugin for SAML services such as Okta or Ping for authentication.
- `IdpTokenAuthPlugin`: An authorization plugin that accepts an AWS IAM Identity Center token or OpenID Connect (OIDC) JSON-based identity tokens (JWT) from any web identity provider linked to AWS IAM Identity Center.
- `JWT`: Use a JSON Web Token (JWT) for authentication.
- `Ping`: Use the PingFederate service for authentication.
- `Okta`: Use the Okta service for authentication.

This parameter is optional.

Port | PortNumber

- **Default Value** – 5439
- **Data Type** – Integer

The number of the TCP port that the Amazon Redshift server uses to listen for client connections.

This parameter is optional.

Prefer_PQ

- **Default Value** – 1
- **Data Type** – Boolean

Specifies whether the driver advertises post-quantum hybrid key-exchange groups in the TLS 1.3 ClientHello. When post-quantum groups are offered and the server also supports them, the server selects one. Otherwise, the handshake falls back to classical key exchange transparently.

- 1 | TRUE: The driver advertises post-quantum hybrid TLS key-exchange groups.
- 0 | FALSE: The driver does not advertise post-quantum groups. Use this setting if a network middlebox blocks TLS handshakes because it does not recognize the post-quantum key-exchange groups the driver offers.

This parameter is optional. It is available in driver versions 2.2.1 and later.

preferred_role

- **Default Value** – None
- **Data Type** – String

The role you want to assume during the connection to Amazon Redshift. It's used for IAM authentication.

This parameter is optional.

Profile

- **Default Value** – None
- **Data Type** – String

The name of the user AWS profile used to authenticate into Amazon Redshift.

- If the Use Instance Profile parameter (the **InstanceProfile** property) is set to 1 | TRUE, that setting takes precedence and the driver uses the Amazon EC2 instance profile instead.
- The default location for the credentials file that contains profiles is `~/.aws/Credentials`. The `AWS_SHARED_CREDENTIALS_FILE` environment variable can be used to point to a different credentials file.

This parameter is optional.

provider_name

- **Default Value** – None
- **Data Type** – String

The authentication provider created by the user using the CREATE IDENTITY PROVIDER query. It's used in native Amazon Redshift authentication.

This parameter is optional.

ProxyHost

- **Default Value** – None
- **Data Type** – String

The host name or IP address of the proxy server that you want to connect through.

This parameter is optional.

ProxyPort

- **Default Value** – None
- **Data Type** – Integer

The number of the port that the proxy server uses to listen for client connections.

This parameter is optional.

ProxyPwd

- **Default Value** – None
- **Data Type** – String

The password that you use to access the proxy server.

This parameter is optional.

ProxyUid

- **Default Value** – None

- **Data Type** – String

The user name that you use to access the proxy server.

This parameter is optional.

ReadOnly

- **Default Value** – 0
- **Data Type** – Boolean

A boolean specifying whether the driver is in read-only mode.

- 1 | TRUE: The connection is in read-only mode, and cannot write to the data store.
- 0 | FALSE: The connection is not in read-only mode, and can write to the data store.

This parameter is optional.

region

- **Default Value** – None
- **Data Type** – String

The AWS region that your cluster is in.

This parameter is optional.

SecretAccessKey

- **Default Value** – None
- **Data Type** – String

The IAM secret key for the user or role. If you set this parameter, you must also set **AccessKeyID**.

This parameter is optional.

SessionToken

- **Default Value** – None

- **Data Type** – String

The temporary IAM session token associated with the IAM role that you are using to authenticate.

This parameter is optional.

Server | HostName | Host

- **Default Value** – None
- **Data Type** – String

The endpoint server to connect to.

This parameter is required.

ssl_insecure

- **Default Value** – 0
- **Data Type** – Boolean

A boolean specifying whether the driver checks the authenticity of the IdP server certificate.

- 1 | TRUE: The driver does not check the authenticity of the IdP server certificate.
- 0 | FALSE: The driver checks the authenticity of the IdP server certificate

This parameter is optional.

SSLMode

- **Default Value** – `verify-full`
- **Data Type** – String

The SSL certificate verification mode to use when connecting to Amazon Redshift. The following values are possible:

- `verify-full`: Connect only using SSL, a trusted certificate authority, and a server name that matches the certificate.
- `verify-ca`: Connect only using SSL and a trusted certificate authority.

- `require`: Connect only using SSL.
- `prefer`: Connect using SSL if available. Otherwise, connect without using SSL.
- `allow`: By default, connect without using SSL. If the server requires SSL connections, then use SSL.
- `disable`: Connect without using SSL.

> **Note**
>
> We recommend using `verify-full` for SSL connections whenever possible.

In driver versions 2.3.0 and later, the default is `verify-full`. In earlier versions, the default was `verify-ca`. If you don't set `SSLMode` and the server certificate doesn't match the host name, the driver logs a warning and connects. If you set `SSLMode`, the driver enforces the mode that you set.

This parameter is optional.

StreamingCursorRows

- **Default Value** – 0
- **Data Type** – Integer

Limits how many rows the driver holds in memory at one time for a forward-only cursor, so that large result sets don't have to load into memory all at once.

Set the property to one of the following values:

- 0: The driver loads the entire result set into memory before returning rows to the application (default). Very large result sets can consume a large amount of memory.
- Positive integer: For a forward-only cursor, the driver keeps only that many rows in memory at a time. It returns those rows to the application, releases them, and then fetches the next batch once the application has consumed the current one.

This option has no effect on scrollable cursors, which always use the in-memory behavior. Note that on Windows, the Amazon Redshift ODBC Driver DSN Setup dialog pre-populates this field with a value of 100.

Streaming cursor mode requires a forward-only cursor. If the Client Side Cursor (CSC) option is enabled, it takes priority and streaming cursor mode is disabled.

> **Note**
>
> `StreamingCursorRows` and `UseDeclareFetch` both batch results instead of loading the entire result set into memory. `UseDeclareFetch` uses a server-side cursor and generally delivers significantly higher throughput, so prefer it when fetch performance is your primary concern. `StreamingCursorRows` does not require transaction wrapping or a server-side cursor; choose it when those properties matter more than peak speed – a forward-only stream with a bounded memory footprint. If `UseDeclareFetch` is enabled, it takes priority and forces `StreamingCursorRows` to 0. For more information, see UseDeclareFetch.

This parameter is optional.

StsConnectionTimeout

- **Default Value** – 0
- **Data Type** – Integer

The maximum wait time for IAM connections, in seconds. If set to 0 or not specified, the driver waits 60 seconds for each AWS STS call.

This parameter is optional.

StsEndpointUrl

- **Default Value** – None
- **Data Type** – String

This option specifies the overriding endpoint used to communicate with the AWS Security Token Service (AWS STS).

This parameter is optional.

token

- **Default Value** – None
- **Data Type** – String

An AWS IAM Identity Center provided access token or an OpenID Connect (OIDC) JSON Web Token (JWT) provided by a web identity provider that's linked with AWS IAM Identity Center. Your application must generate this token by authenticating the user of your application with AWS IAM Identity Center or an identity provider linked with AWS IAM Identity Center.

This parameter works with `IdpTokenAuthPlugin`.

token_type

- **Default Value** – None
- **Data Type** – String

The type of token that is being used in `IdpTokenAuthPlugin`.

You can specify the following values:

ACCESS_TOKEN

Enter this if you use an AWS IAM Identity Center provided access token.

EXT_JWT

Enter this if you use an OpenID Connect (OIDC) JSON Web Token (JWT) provided by a web-based identity provider that's integrated with AWS IAM Identity Center.

This parameter works with `IdpTokenAuthPlugin`.

UID | User | LogonID

- **Default Value** – None
- **Data Type** – String

The user name that you use to access the Amazon Redshift server.

This parameter is required if you use database authentication.

UseDeclareFetch

- **Default Value** – 0
- **Data Type** – Boolean

When a query returns a result set too large to hold in memory, enabling this option lets the driver process it in fixed-size batches instead of buffering the entire result at once, keeping the client's memory footprint bounded and predictable regardless of how many rows the query returns. It is a boolean specifying whether the driver uses Declare/Fetch mode, returning a set number of rows at a time. To set the number of rows returned per batch, use the `Fetch` option. For more information, see Fetch.

- 1 | TRUE: The driver uses Declare/Fetch mode and returns the number of rows specified by the `Fetch` option at a time.
- 0 | FALSE: The driver returns the entire query result at once.

This parameter is optional.

> **Note**
>
> If `UseDeclareFetch` is enabled, the driver disables streaming cursor mode and overrides `StreamingCursorRows` to 0, regardless of the value set in your DSN or connection string. For more information, see StreamingCursorRows.

UseUnicode

- **Default Value** – 0
- **Data Type** – Boolean

A boolean specifying whether the driver returns Redshift data as Unicode or regular SQL types.

- 1 | TRUE: The Driver returns wide SQL type for character data type.
  - SQL_WCHAR is returned instead of SQL_CHAR.
  - SQL_WVARCHAR is returned instead of SQL_VARCHAR.

  - SQL_WLONGVARCHAR is returned instead of SQL_LONGVARCHAR.
- 0 | FALSE: The driver returns normal SQL type for character data type.
  - SQL_CHAR is returned instead of SQL_WCHAR.
  - SQL_VARCHAR is returned instead of SQL_WVARCHAR.
  - SQL_LONGVARCHAR is returned instead of SQL_WLONGVARCHAR.

This parameter is optional. It is available in driver versions 2.1.15 and later.

web_identity_token

- **Default Value** – None
- **Data Type** – String

The OAUTH token that is provided by the identity provider. It's used in the JWT plugin.

This parameter is required if you set the **plugin_name** parameter to BasicJwtCredentialsProvider.

Previous ODBC driver versions

Download a previous version of the Amazon Redshift ODBC driver version 2.x only if your tool requires a specific version of the driver.

Use previous ODBC driver versions for Microsoft Windows

The following are the previous versions of the Amazon Redshift ODBC driver version 2.x for Microsoft Windows:

- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.4.0/AmazonRedshiftODBC64-2.2.4.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.3.0/AmazonRedshiftODBC64-2.2.3.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.2.0/AmazonRedshiftODBC64-2.2.2.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.1.0/AmazonRedshiftODBC64-2.2.1.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.0.0/AmazonRedshiftODBC64-2.2.0.0.msi

- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.17.0/AmazonRedshiftODBC64-2.1.17.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.16.0/AmazonRedshiftODBC64-2.1.16.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.15.0/AmazonRedshiftODBC64-2.1.15.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.14.0/AmazonRedshiftODBC64-2.1.14.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.13.0/AmazonRedshiftODBC64-2.1.13.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.12.0/AmazonRedshiftODBC64-2.1.12.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.11.0/AmazonRedshiftODBC64-2.1.11.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.10.0/AmazonRedshiftODBC64-2.1.10.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.9.0/AmazonRedshiftODBC64-2.1.9.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.9.0/AmazonRedshiftODBC64-2.1.9.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.8.0/AmazonRedshiftODBC64-2.1.8.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.7.0/AmazonRedshiftODBC64-2.1.7.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.6.0/AmazonRedshiftODBC64-2.1.6.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.4.0/AmazonRedshiftODBC64-2.1.4.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.3.0/AmazonRedshiftODBC64-2.1.3.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.2.0/AmazonRedshiftODBC64-2.1.2.0.msi

- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.1.0/AmazonRedshiftODBC64-2.1.1.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.0.0/AmazonRedshiftODBC64-2.1.0.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.1.0/AmazonRedshiftODBC64-2.0.1.0.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.11/AmazonRedshiftODBC64-2.0.0.11.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.9/AmazonRedshiftODBC64-2.0.0.9.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.8/AmazonRedshiftODBC64-2.0.0.8.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.7/AmazonRedshiftODBC64-2.0.0.7.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.6/AmazonRedshiftODBC64-2.0.0.6.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.5/AmazonRedshiftODBC64-2.0.0.5.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.3/AmazonRedshiftODBC64-2.0.0.3.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.1/AmazonRedshiftODBC64-2.0.0.1.msi

Use previous ODBC driver versions for Linux

The following are the previous versions of the Amazon Redshift ODBC driver version 2.x for Linux:

- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.4.0/AmazonRedshiftODBC-64-bit-2.2.4.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.4.0/AmazonRedshiftODBC-64-bit-2.2.4.0.aarch64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.4.0/AmazonRedshiftODBC-64-bit-2.2.4.0.amd64.deb

- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.4.0/AmazonRedshiftODBC-64-bit-2.2.4.0.arm64.deb
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.3.0/AmazonRedshiftODBC-64-bit-2.2.3.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.3.0/AmazonRedshiftODBC-64-bit-2.2.3.0.aarch64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.3.0/AmazonRedshiftODBC-64-bit-2.2.3.0.amd64.deb
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.3.0/AmazonRedshiftODBC-64-bit-2.2.3.0.arm64.deb
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.2.0/AmazonRedshiftODBC-64-bit-2.2.2.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.2.0/AmazonRedshiftODBC-64-bit-2.2.2.0.aarch64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.2.0/AmazonRedshiftODBC-64-bit-2.2.2.0.amd64.deb
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.2.0/AmazonRedshiftODBC-64-bit-2.2.2.0.arm64.deb
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.1.0/AmazonRedshiftODBC-64-bit-2.2.1.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.1.0/AmazonRedshiftODBC-64-bit-2.2.1.0.aarch64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.1.0/AmazonRedshiftODBC-64-bit-2.2.1.0.amd64.deb
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.1.0/AmazonRedshiftODBC-64-bit-2.2.1.0.arm64.deb
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.0.0/AmazonRedshiftODBC-64-bit-2.2.0.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.0.0/AmazonRedshiftODBC-64-bit-2.2.0.0.aarch64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.0.0/AmazonRedshiftODBC-64-bit-2.2.0.0.amd64.deb

- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.0.0/AmazonRedshiftODBC-64-bit-2.2.0.0.arm64.deb
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.17.0/AmazonRedshiftODBC-64-bit-2.1.17.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.17.0/AmazonRedshiftODBC-64-bit-2.1.17.0.aarch64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.16.0/AmazonRedshiftODBC-64-bit-2.1.16.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.16.0/AmazonRedshiftODBC-64-bit-2.1.16.0.aarch64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.15.0/AmazonRedshiftODBC-64-bit-2.1.15.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.15.0/AmazonRedshiftODBC-64-bit-2.1.15.0.aarch64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.14.0/AmazonRedshiftODBC-64-bit-2.1.14.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.14.0/AmazonRedshiftODBC-64-bit-2.1.14.0.aarch64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.13.0/AmazonRedshiftODBC-64-bit-2.1.13.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.13.0/AmazonRedshiftODBC-64-bit-2.1.13.0.aarch64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.12.0/AmazonRedshiftODBC-64-bit-2.1.12.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.12.0/AmazonRedshiftODBC-64-bit-2.1.12.0.aarch64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.11.0/AmazonRedshiftODBC-64-bit-2.1.11.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.11.0/AmazonRedshiftODBC-64-bit-2.1.11.0.aarch64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.10.0/AmazonRedshiftODBC-64-bit-2.1.10.0.x86_64.rpm

- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.9.0/AmazonRedshiftODBC-64-bit-2.1.9.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.8.0/AmazonRedshiftODBC-64-bit-2.1.8.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.7.0/AmazonRedshiftODBC-64-bit-2.1.7.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.6.0/AmazonRedshiftODBC-64-bit-2.1.6.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.4.0/AmazonRedshiftODBC-64-bit-2.1.4.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.3.0/AmazonRedshiftODBC-64-bit-2.1.3.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.2.0/AmazonRedshiftODBC-64-bit-2.1.2.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.1.0/AmazonRedshiftODBC-64-bit-2.1.1.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.0.0/AmazonRedshiftODBC-64-bit-2.1.0.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.1.0/AmazonRedshiftODBC-64-bit-2.0.1.0.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.11/AmazonRedshiftODBC-64-bit-2.0.0.11.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.9/AmazonRedshiftODBC-64-bit-2.0.0.9.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.8/AmazonRedshiftODBC-64-bit-2.0.0.8.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.7/AmazonRedshiftODBC-64-bit-2.0.0.7.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.6/AmazonRedshiftODBC-64-bit-2.0.0.6.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.5/AmazonRedshiftODBC-64-bit-2.0.0.5.x86_64.rpm

- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.3/AmazonRedshiftODBC-64-bit-2.0.0.3.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.0.0.1/AmazonRedshiftODBC-64-bit-2.0.0.1.x86_64.rpm

Use previous ODBC driver versions for Apple macOS

The following are the previous versions of the Amazon Redshift ODBC driver version 2.x for Apple macOS:

- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.4.0/AmazonRedshiftODBC-64-bit.2.2.4.0.universal.pkg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.3.0/AmazonRedshiftODBC-64-bit.2.2.3.0.universal.pkg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.2.0/AmazonRedshiftODBC-64-bit.2.2.2.0.universal.pkg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.1.0/AmazonRedshiftODBC-64-bit.2.2.1.0.universal.pkg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.2.0.0/AmazonRedshiftODBC-64-bit.2.2.0.0.universal.pkg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.17.0/AmazonRedshiftODBC-64-bit.2.1.17.0.universal.pkg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.16.0/AmazonRedshiftODBC-64-bit.2.1.16.0.universal.pkg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.15.0/AmazonRedshiftODBC-64-bit.2.1.15.0.universal.pkg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.14.0/AmazonRedshiftODBC-64-bit.2.1.14.0.universal.pkg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.13.0/AmazonRedshiftODBC-64-bit.2.1.13.0.universal.pkg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/2.1.12.0/AmazonRedshiftODBC-64-bit.2.1.12.0.universal.pkg

Differences between the 2.x and 1.x versions of the ODBC driver

This section describes the differences between the ODBC 1.x driver and the ODBC 2.x driver, and provides guidance about how to migrate to the 2.x driver. It describes the changes that might affect your application and how to address them.

Key differences to address

The following actions resolve the most common migration issues. Most applications need only a subset of these.

DSN and connection string

- **Review your DSN for driver options that might have been removed or renamed.** Some ODBC 1.x driver options are not supported in the ODBC 2.x driver. Others have been renamed. See Connection string and DSN options for details.
- **Set `UseUnicode=true` if your application depends on wide character type codes (`SQL_WVARCHAR`, `SQL_WCHAR`).** In the ODBC 2.x driver, `UseUnicode` defaults to `false`, which reports narrow character types.
- **Move logging settings to the `[ODBC]` section.** On Linux and macOS, `LogLevel` and `LogPath` must be set in the global `[ODBC]` section of `odbc.ini`, not in an individual DSN section.

Query and schema

- **Include `EXTERNAL TABLE` in `SQLTables` type filters.** In ODBC 2.x, both Amazon Redshift Spectrum tables and datashare tables are reported as `EXTERNAL TABLE`. If you filter `SQLTables` by type, add `EXTERNAL TABLE` to keep seeing these objects. Alternatively, set the `EnableTableTypes` option to 0 to normalize the detailed table type information into the generic TABLE and VIEW table types, as described in the following item.
- **Detailed table types are enabled by default.** ODBC 2.x enables the `EnableTableTypes` option by default. The 1.x driver disabled this option by default. With the 2.x default, `SQLTables` reports detailed types such as SYSTEM TABLE, SYSTEM VIEW, EXTERNAL TABLE, and LOCAL TEMPORARY. To report every table as the generic type TABLE and every view as VIEW, set `EnableTableTypes` to 0.
- **Cast `INTERVAL` to `VARCHAR` for applications that do not support the interval data type.** Some clients do not support ODBC interval types. For those clients, cast the column in your query: `SELECT col::VARCHAR FROM ...`

Application code

- **Verify query timeout settings.** ODBC 2.x correctly enforces `SQL_ATTR_QUERY_TIMEOUT` per the ODBC specification. ODBC 1.x silently ignored this setting. Long-running queries that previously succeeded might now fail with a timeout error. Review and adjust your timeout values as needed.
- **Provide the data length for data-at-execution parameters.** 2.x reports `SQL_NEED_LONG_DATA_LEN` as `Y` (1.x reported `N`). Applications that bind `SQL_DATA_AT_EXEC` parameters must now supply the total data length up front in `StrLen_or_IndPtr`.

Connection string and DSN options

The following table shows ODBC 1.x driver options that have either been renamed or have direct equivalents in the ODBC 2.x driver.

1.x option	2.x equivalent	Notes
`MaxLongVarChar`	`MaxLongVarcharSize`	Default changed from 8190 to 65535.
`ConnectionTimeout`	`LoginTimeout`	Same connection timeout, renamed. Defaults to 0 (no timeout).
`VpcEndpointUrl`	`vpc_endpoint_url`	
`SSLCertPath`	`TrustStore` or `CaFile`	Path to a CA certificate used to verify the server. On Windows, set this in the Trust Store field of the DSN setup dialog; the dialog has no `CaFile` field. If both are set, `TrustStore` takes priority.

The following 1.x options are not supported in the current 2.x driver. The 2.x driver ignores them, so they do not affect your connection. Removing them from your DSN is optional but recommended to avoid confusion.

- `SingleRowMode` – to limit client memory, use `StreamingCursorRows` or `UseDeclareFetch` instead.

- `UseSystemTrustStore` – not supported. On Windows, the 1.x driver could validate the server certificate against the Windows system certificate store. The 2.x driver validates against a CA certificate file: it uses the bundled Amazon Redshift root certificate by default, or the file you specify in `TrustStore` or `CaFile`.
- `TextAsLongVarchar`, `CheckCertRevocation`, `EnableAwsSdkLogs`, `UseLogPrefix`, `Locale`, `UseMultipleStatements`, `EnforceSingleStatement` – no equivalent in the current release. The Amazon Redshift team is evaluating equivalents or alternatives for these options in future releases.

For the full list of supported 2.x options, see ODBC driver options.

Data type changes

The following table shows Amazon Redshift data type mappings that changed in ODBC 2.x to comply with the ODBC specification. Most applications are unaffected because they bind columns by index or by name rather than by type code.

Amazon Redshift data type	ODBC 1.x	ODBC 2.x
`DOUBLE PRECISION`	`SQL_FLOAT` (6)	`SQL_DOUBLE` (8)
`INTERVAL YEAR TO MONTH`	`SQL_VARCHAR` (12), text	`SQL_INTERVAL_YEAR_TO_MONTH` (107), interval struct
`INTERVAL DAY TO SECOND`	`SQL_VARCHAR` (12), text	`SQL_INTERVAL_DAY_TO_SECOND` (110), interval struct
`VARCHAR` / `CHAR` / `TEXT`	Wide types (`SQL_WVARCHAR`, `SQL_WCHAR`, `SQL_WLONGVARCHAR`); default `UseUnicode=true`	Narrow types (`SQL_VARCHAR`, `SQL_CHAR`, `SQL_LONGVARCHAR`). Set `UseUnicode=true` to restore wide types.
`GEOMETRY` / `GEOGRAPHY` (as `SQL_C_BINARY`)	Hex-encoded ASCII string	Raw binary bytes

The COLUMN_SIZE values returned by SQLColumns also changed for GEOMETRY, GEOGRAPHY, and SUPER data types, which now return NULL to indicate unsized columns per the ODBC specification. Applications that allocate buffers based on COLUMN_SIZE must handle NULL by using a default buffer size.

Post-migration troubleshooting

The following table describes common issues you might encounter after migrating and how to resolve them.

Symptom	Cause	What to do
External tables not visible in schema browser	Spectrum and datashare tables are reported as EXTERNAL TABLE	Include EXTERNAL TABLE in your SQLTables type filter, or set EnableTableTypes to 0 to normalize the detailed table types into the generic TABLE and VIEW types.
pyodbc errors on interval columns	pyodbc does not support ODBC interval types	Cast intervals to VARCHAR in queries.
Character data displays as unexpected characters (mojibake)	UseUnicode default changed to false. Applicati ons expecting wide-char acter (UTF-16) data might misinterpret narrow-character bytes as wide pairs, producing garbled output. The data itself is unchanged.	Set UseUnicode=true in your DSN, or update your application to bind columns as SQL_C_CHAR instead of SQL_C_WCHAR .
Long-running queries return timeout errors	SQL_ATTR_QUERY_TIM EOUT is now enforced. ODBC 1.x silently ignored this setting.	Increase or remove QueryTimeout from your DSN, or set SQL_ATTR_ QUERY_TIMEOUT to 0 in your application.
Setting SQL_ATTR_ CURRENT_CATALOG returns HY011	The attribute cannot be set on an open connectio n. Switching databases on	Set the attribute before connecting, or close and reopen the connection on the target database.

Symptom	Cause	What to do
	an open connection is not supported.	
Connection hangs in dual-stack environments	Limitation in versions <= 2.1.16	Upgrade to 2.1.17 or later.
Error message parsing returns unexpected results	Message format changed; SQLSTATE codes unchanged	Parse SQLSTATE codes instead of message text.
Data-at-execution parameter binding behaves differently	`SQL_NEED_LONG_DATA_LEN` changed from FALSE to TRUE; applications using `SQL_DATA_AT_EXEC` must now provide the total data length up front	Set the length value in `StrLen_or_IndPtr` when binding `SQL_DATA_AT_EXEC` parameters.
Driver logs are not generated	`LogLevel` and `LogPath` must be in the `[ODBC]` global section of `odbc.ini`, not in individual DSN sections	Move logging settings from your DSN to the `[ODBC]` section.

Upgrading from an older ODBC 2.x version

If you are already running an older ODBC 2.x version, upgrade to the latest release. For a complete list of enhancements and bug fixes across all versions, see the Amazon Redshift ODBC driver change log on GitHub.

More information

- Configuring a connection for ODBC driver version 2.x for Amazon Redshift
- ODBC driver options
- Amazon Redshift ODBC driver change log

Configuring an ODBC driver version 1.x connection

You can use an ODBC connection to connect to your Amazon Redshift cluster from many third-party SQL client tools and applications. To do this, set up the connection on your client computer or Amazon EC2 instance. If your client tool supports JDBC, you might choose to use that type of connection rather than ODBC due to the ease of configuration that JDBC provides. However, if your client tool doesn't support JDBC, follow the steps in this section to configure an ODBC connection.

Amazon Redshift provides 64-bit ODBC drivers for Linux, Windows, and macOS operating systems. The 32-bit ODBC drivers are discontinued. Further updates will not be released, except for urgent security patches.

For the latest information about ODBC driver functionality and prerequisites, see Amazon Redshift ODBC driver release notes.

For installation and configuration information for Amazon Redshift ODBC drivers, see the Amazon Redshift ODBC connector installation and configuration guide.

Topics

- Getting the ODBC URL
- Using an Amazon Redshift ODBC driver on Microsoft Windows
- Using an Amazon Redshift ODBC driver on Linux
- Using an Amazon Redshift ODBC driver on macOS
- ODBC driver options
- Previous ODBC driver versions

Getting the ODBC URL

Amazon Redshift displays the ODBC URL for your cluster in the Amazon Redshift console. This URL contains the information to set up the connection between your client computer and the database.

An ODBC URL has the following format:

```
Driver={driver};Server=endpoint;Database=database_name;UID=user_name;PWD=passwor
```

The fields of the format shown preceding have the following values.

Field	Value
`Driver`	The name of the 64-bit ODBC driver to use: **Amazon Redshift (x64)**.

Field	Value
	The name of the 32-bit ODBC driver: **Amazon Redshift (x86)**.
`Server`	The endpoint of the Amazon Redshift cluster.
`Database`	The database that you created for your cluster.
`UID`	The user name of a user account that has permission to connect to the database. This value is a database permission, not an Amazon Redshift permission, although you can use the admin user account that you set up when you launched the cluster.
`PWD`	The password for the user account to connect to the database.
`Port`	The port number that you specified when you launched the cluster. If you have a firewall, ensure that this port is open for you to use.

The fields in the preceding tables can contain the following special characters:

```
[]{}(),;?*=!@
```

If you use these special characters you must enclose the value in curly braces. For example, the password value `Your;password123` in a connection string is represented as `PWD={Your;password123};`.

Since `Field=value` pairs are separated by semicolon, the combination of `}` and `;` with any number of spaces in between is considered the end of a `Field={value};` pair. We recommend you avoid the sequence `};` in your field values. For example, if you set your password value as `PWD={This is a passwor} ;d};`, your password would be `This is a passwor} ;` and the URL would error out.

The following is an example ODBC URL.

```
Driver={Amazon Redshift (x64)};
                    Server=examplecluster.abc123xyz789.us-
west-2.redshift.amazonaws.com;
                    Database=dev;
                    UID=adminuser;
```

```
PWD=insert_your_admin_user_password_here;
Port=5439
```

For information about how to get your ODBC connection, see Finding your cluster connection string.

Using an Amazon Redshift ODBC driver on Microsoft Windows

You install the Amazon Redshift ODBC driver on client computers accessing an Amazon Redshift data warehouse. Each computer where you install the driver must meet a list of minimum system requirements. For information about minimum system requirements, see the Amazon Redshift ODBC connector installation and configuration guide.

Topics

- Downloading and installing the Amazon Redshift ODBC driver
- Creating a system DSN entry for an ODBC connection

Downloading and installing the Amazon Redshift ODBC driver

Use the following procedure to download the Amazon Redshift ODBC drivers for Windows operating systems. Only use a driver other than these if you're running a third-party application that is certified for use with Amazon Redshift and that requires a specific driver.

To install the ODBC driver

1. Download one of the following, depending on the system architecture of your SQL client tool or application:

   - 64-bit ODBC driver version 1.6.5

     The name for this driver is Amazon Redshift (x64).

   - 32-bit ODBC driver version 1.4.52

     The name for this driver is Amazon Redshift (x86). The 32-bit ODBC drivers are discontinued. Further updates will not be released, except for urgent security patches.

> **Note**
>
> Download the MSI package that corresponds to the system architecture of your SQL client tool or application. For example, if your SQL client tool is 64-bit, install the 64-bit driver.

Then download and review the Amazon Redshift ODBC and JDBC driver license agreement.

2. Double-click the .msi file, and then follow the steps in the wizard to install the driver.

Creating a system DSN entry for an ODBC connection

After you download and install the ODBC driver, add a data source name (DSN) entry to the client computer or Amazon EC2 instance. SQL client tools use this data source to connect to the Amazon Redshift database.

We recommend that you create a system DSN instead of a user DSN. Some applications load the data using a different user account. These applications might not be able to detect user DSNs that are created under another user account.

> **Note**
>
> For authentication using AWS Identity and Access Management (IAM) credentials or identity provider (IdP) credentials, additional steps are required. For more information, see Step 5: Configure a JDBC or ODBC connection to use IAM credentials.

For information about how to create a system DSN entry, see the Amazon Redshift ODBC connector installation and configuration guide.

To create a system DSN entry for an ODBC connection on Windows

1. In the **Start** menu, open **ODBC Data Sources**.

   Make sure that you choose the ODBC Data Source Administrator that has the same bitness as the client application that you are using to connect to Amazon Redshift.

2. In the **ODBC Data Source Administrator**, choose the **Driver** tab and locate the driver folder:

- **Amazon Redshift ODBC Driver (64-bit)**
- **Amazon Redshift ODBC Driver (32-bit)**

3. Choose the **System DSN** tab to configure the driver for all users on the computer, or the **User DSN** tab to configure the driver for your user account only.
4. Choose **Add**. The **Create New Data Source** window opens.
5. Choose the **Amazon Redshift** ODBC driver, and then choose **Finish**. The **Amazon Redshift ODBC Driver DSN Setup** window opens.
6. Under **Connection Settings**, enter the following information:

   Data source name

   Enter a name for the data source. You can use any name that you want to identify the data source later when you create the connection to the cluster. For example, if you followed the *Amazon Redshift Getting Started Guide*, you might type `exampleclusterdsn` to make it easy to remember the cluster that you associate with this DSN.

   Server

   Specify the endpoint for your Amazon Redshift cluster. You can find this information in the Amazon Redshift console on the cluster's details page. For more information, see Configuring connections in Amazon Redshift.

   Port

   Enter the port number that the database uses. Use the port that the cluster was configured to use when it was launched or modified.

   Database

   Enter the name of the Amazon Redshift database. If you launched your cluster without specifying a database name, enter *dev*. Otherwise, use the name that you chose during the launch process. If you followed the *Amazon Redshift Getting Started Guide*, enter *dev*.
7. Under **Authentication**, specify the configuration options to configure standard or IAM authentication. For information about authentication options, see "Configuring Authentication on Windows" in *Amazon Redshift ODBC Connector Installation and Configuration Guide*.
8. Under **SSL Settings**, specify a value for the following:

SSL authentication

Choose a mode for handling Secure Sockets Layer (SSL). In a test environment, you might use `prefer`. However, for production environments and when secure data exchange is required, use `verify-ca` or `verify-full`. For more information about using SSL on Windows, see "Configuring SSL Verification on Windows" in *Amazon Redshift ODBC Connector Installation and Configuration Guide*.

9. Under **Additional Options**, specify options on how to return query results to your SQL client tool or application. For more information, see "Configuring Additional Options on Windows" in *Amazon Redshift ODBC Connector Installation and Configuration Guide*.
10. In **Logging Options**, specify values for the logging option. For more information, see "Configuring Logging Options on Windows" in *Amazon Redshift ODBC Connector Installation and Configuration Guide*.

    Then choose **OK**.
11. Under **Data Type Options**, specify values for data types. For more information, see "Configuring Data Type Options on Windows" in *Amazon Redshift ODBC Connector Installation and Configuration Guide*.

    Then choose **OK**.
12. Choose **Test**. If the client computer can connect to the Amazon Redshift database, you see the following message: **Connection successful**.

    If the client computer fails to connect to the database, you can troubleshoot possible issues. For more information, see Troubleshooting connection issues in Amazon Redshift.
13. Configure TCP keepalives on Windows to prevent connections from timing out. For information about how to configure TCP keepalives on Windows, see *Amazon Redshift ODBC Connector Installation and Configuration Guide*.
14. To help troubleshooting, configure logging. For information about how to configure logging on Windows, see *Amazon Redshift ODBC Connector Installation and Configuration Guide*.

Using an Amazon Redshift ODBC driver on Linux

You install the Amazon Redshift ODBC driver on client computers accessing an Amazon Redshift data warehouse. Each computer where you install the driver must meet a list of minimum system

requirements. For information about minimum system requirements, see the Amazon Redshift ODBC connector installation and configuration guide.

Topics

- Downloading and installing the Amazon Redshift ODBC driver
- Using an ODBC driver manager to configure the driver

Downloading and installing the Amazon Redshift ODBC driver

Use the steps in this section to download and install the Amazon Redshift ODBC drivers on a supported Linux distribution. The installation process installs the driver files in the following directories:

- `/opt/amazon/redshiftodbc/lib/64` (for the 64-bit driver)
- `/opt/amazon/redshiftodbc/ErrorMessages`
- `/opt/amazon/redshiftodbc/Setup`
- `/opt/amazon/redshiftodbc/lib/32` (for the 32-bit driver)

To install the Amazon Redshift ODBC driver

1. Download one of the following, depending on the system architecture of your SQL client tool or application:

   - 64-bit RPM driver version 1.6.5
   - 64-bit Debian driver version 1.6.5
   - 32-bit driver version 1.4.52

   The name for each of these drivers is Amazon Redshift ODBC driver. The 32-bit ODBC drivers are discontinued. Further updates will not be released, except for urgent security patches.

   > **Note**
   >
   > Download the package that corresponds to the system architecture of your SQL client tool or application. For example, if your client tool is 64-bit, install a 64-bit driver.

Then download and review the Amazon Redshift ODBC and JDBC driver license agreement.

2. Go to the location where you downloaded the package, and then run one of the following commands. Use the command that corresponds to your Linux distribution.

   - On RHEL and CentOS operating systems, run the following command.

     ```
     yum -nogpgcheck localinstall RPMFileName
     ```

     Replace *RPMFileName* with the RPM package file name. For example, the following command demonstrates installing the 64-bit driver.

     ```
     yum -nogpgcheck localinstall AmazonRedshiftODBC-64-bit-1.x.xx.xxxx-x.x86_64.rpm
     ```

   - On SLES, run the following command.

     ```
     zypper install RPMFileName
     ```

     Replace *RPMFileName* with the RPM package file name. For example, the following command demonstrates installing the 64-bit driver.

     ```
     zypper install AmazonRedshiftODBC-1.x.x.xxxx-x.x86_64.rpm
     ```

   - On Debian, run the following command.

     ```
     sudo apt install ./DEBFileName.deb
     ```

     Replace *DEBFileName.deb* with the Debian package file name. For example, the following command demonstrates installing the 64-bit driver.

     ```
     sudo apt install ./AmazonRedshiftODBC-1.x.x.xxxx-x.x86_64.deb
     ```

> **Important**
>
> When you have finished installing the drivers, configure them for use on your system. For more information on driver configuration, see Using an ODBC driver manager to configure the driver.

Using an ODBC driver manager to configure the driver

On Linux operating systems, you use an ODBC driver manager to configure the ODBC connection settings. ODBC driver managers use configuration files to define and configure ODBC data sources and drivers. The ODBC driver manager that you use depends on the operating system that you use. For Linux, it's unixODBC driver manager.

For more information about the supported ODBC driver managers to configure the Amazon Redshift ODBC drivers, see Using an Amazon Redshift ODBC driver on Linux for Linux operating systems. Also, see "Specifying ODBC Driver Managers on Non- Windows Machines" in the Amazon Redshift ODBC connector installation and configuration guide.

Three files are required for configuring the Amazon Redshift ODBC driver: `amazon.redshiftodbc.ini`, `odbc.ini`, and `odbcinst.ini`.

If you installed to the default location, the `amazon.redshiftodbc.ini` configuration file is located in one of the following directories:

- `/opt/amazon/redshiftodbc/lib/64` (for the 64-bit driver on Linux operating systems)
- `/opt/amazon/redshiftodbc/lib/32` (for the 32-bit driver on Linux operating systems)

Additionally, under `/opt/amazon/redshiftodbc/Setup` on Linux, there are sample `odbc.ini` and `odbcinst.ini` files. You can use these files as examples for configuring the Amazon Redshift ODBC driver and the data source name (DSN).

We don't recommend using the Amazon Redshift ODBC driver installation directory for the configuration files. The sample files in the `Setup` directory are for example purposes only. If you reinstall the Amazon Redshift ODBC driver at a later time, or upgrade to a newer version, the installation directory is overwritten. You then lose any changes that you might have made to those files.

To avoid this, copy the `amazon.redshiftodbc.ini` file to a directory other than the installation directory. If you copy this file to the user's home directory, add a period (.) to the beginning of the file name to make it a hidden file.

For the `odbc.ini` and `odbcinst.ini` files, either use the configuration files in the user's home directory or create new versions in another directory. By default, your Linux operating system should have an `odbc.ini` file and an `odbcinst.ini` file in the user's home directory (`/home/$USER` or `~/`). These default files are hidden files, which is indicated by the dot (.) in front of each file name. These files display only when you use the `-a` flag to list the directory contents.

Whichever option you choose for the `odbc.ini` and `odbcinst.ini` files, modify the files to add driver and DSN configuration information. If you create new files, you also need to set environment variables to specify where these configuration files are located.

By default, ODBC driver managers are configured to use hidden versions of the `odbc.ini` and `odbcinst.ini` configuration files (named `.odbc.ini` and `.odbcinst.ini`) located in the home directory. They also are configured to use the `amazon.redshiftodbc.ini` file in the `/lib` subfolder of the driver installation directory. If you store these configuration files elsewhere, set the environment variables described following so that the driver manager can locate the files. For more information, see "Specifying the Locations of the Driver Configuration Files" in the Amazon Redshift ODBC connector installation and configuration guide.

Creating a data source name on Linux operating systems

When connecting to your data store using a data source name (DSN), configure the `odbc.ini` file to define DSNs. Set the properties in the `odbc.ini` file to create a DSN that specifies the connection information for your data store.

For information about how to configure the `odbc.ini` file, see "Creating a Data Source Name on a Non-Windows Machine" in the Amazon Redshift ODBC connector installation and configuration guide

Use the following format on Linux operating systems.

```
[ODBC Data Sources]
driver_name=dsn_name

[dsn_name]
Driver=path/driver_file
```

```
Host=cluster_endpoint
Port=port_number
Database=database_name
locale=locale
```

The following example shows the configuration for odbc.ini with the 64-bit ODBC driver on Linux operating systems.

```
[ODBC Data Sources]
Amazon_Redshift_x64=Amazon Redshift (x64)

[Amazon Redshift (x64)]
Driver=/opt/amazon/redshiftodbc/lib/64/libamazonredshiftodbc64.so
Host=examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com
Port=5932
Database=dev
locale=en-US
```

The following example shows the configuration for odbc.ini with the 32-bit ODBC driver on Linux operating systems.

```
[ODBC Data Sources]
Amazon_Redshift_x32=Amazon Redshift (x86)

[Amazon Redshift (x86)]
Driver=/opt/amazon/redshiftodbc/lib/32/libamazonredshiftodbc32.so
Host=examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com
Port=5932
Database=dev
locale=en-US
```

Configuring a connection without a DSN on Linux operating systems

To connect to your data store through a connection that doesn't have a DSN, define the driver in the `odbcinst.ini` file. Then provide a DSN-less connection string in your application.

For information about how to configure the `odbcinst.ini` file in this case, see "Configuring a DSN-less Connection on a Non-Windows Machine" in the Amazon Redshift ODBC connector installation and configuration guide.

Use the following format on Linux operating systems.

```
[ODBC Drivers]
driver_name=Installed
...

[driver_name]
Description=driver_description
Driver=path/driver_file

...
```

The following example shows the `odbcinst.ini` configuration for the 64-bit driver installed in the default directories on Linux operating systems.

```
[ODBC Drivers]
Amazon Redshift (x64)=Installed

[Amazon Redshift (x64)]
Description=Amazon Redshift ODBC Driver (64-bit)
Driver=/opt/amazon/redshiftodbc/lib/64/libamazonredshiftodbc64.so
```

The following example shows the `odbcinst.ini` configuration for the 32-bit driver installed in the default directories on Linux operating systems.

```
[ODBC Drivers]
Amazon Redshift (x86)=Installed

[Amazon Redshift (x86)]
Description=Amazon Redshift ODBC Driver (32-bit)
Driver=/opt/amazon/redshiftodbc/lib/32/libamazonredshiftodbc32.so
```

Configuring environment variables

Use the correct ODBC driver manager to load the correct driver. To do this, set the library path environment variable. For more information, see "Specifying ODBC Driver Managers on Non-Windows Machines" in the Amazon Redshift ODBC connector installation and configuration guide.

By default, ODBC driver managers are configured to use hidden versions of the `odbc.ini` and `odbcinst.ini` configuration files (named `.odbc.ini` and `.odbcinst.ini`) located in the home directory. They also are configured to use the `amazon.redshiftodbc.ini` file in the `/lib` subfolder of the driver installation directory. If you store these configuration files elsewhere, the

environment variables so that the driver manager can locate the files. For more information, see "Specifying the Locations of the Driver Configuration Files" in *Amazon Redshift ODBC Connector Installation and Configuration Guide*.

Configuring connection features

You can configure the following connection features for your ODBC setting:

- Configure the ODBC driver to provide credentials and authenticate the connection to the Amazon Redshift database.
- Configure the ODBC driver to connect to a socket enabled with Secure Sockets Layer (SSL), if you are connecting to an Amazon Redshift server that has SSL enabled.
- Configure the ODBC driver to connect to Amazon Redshift through a proxy server.
- Configure the ODBC driver to use a query processing mode to prevent queries from consuming too much memory.
- Configure the ODBC driver to pass IAM authentication processes through a proxy server.
- Configure the ODBC driver to use TCP keepalives to prevent connections from timing out.

For information about these connection features, see the Amazon Redshift ODBC connector installation and configuration guide.

Using an Amazon Redshift ODBC driver on macOS

You install the driver on client computers accessing an Amazon Redshift data warehouse. Each computer where you install the driver must meet a list of minimum system requirements. For information about minimum system requirements, see the Amazon Redshift ODBC connector installation and configuration guide.

Topics

- Downloading and installing the Amazon Redshift ODBC driver
- Use an ODBC driver manager to configure the driver

Downloading and installing the Amazon Redshift ODBC driver

Use the steps in this section to download and install the Amazon Redshift ODBC driver on a supported version of macOS. The installation process installs the driver files in the following directories:

- `/opt/amazon/redshift/lib/universal`
- `/opt/amazon/redshift/ErrorMessages`
- `/opt/amazon/redshift/Setup`

To install the Amazon Redshift ODBC driver on macOS

1. To install the Amazon Redshift ODBC driver on macOS, download the macOS driver version 1.6.5.

   Then download and review the Amazon Redshift ODBC and JDBC driver license agreement.
2. Double-click **AmazonRedshiftODBC.pkg** to run the installer.
3. Follow the steps in the installer to complete the driver installation process. To perform the installation, agree to the terms of the license agreement.

> **Important**
>
> When you have finished installing the driver, configure it for use on your system. For more information on driver configuration, see Use an ODBC driver manager to configure the driver.

Use an ODBC driver manager to configure the driver

On macOS operating systems, you use an ODBC driver manager to configure the ODBC connection settings. ODBC driver managers use configuration files to define and configure ODBC data sources and drivers. The ODBC driver manager that you use depends on the operating system that you use. For a macOS operation system, it's the iODBC driver manager.

For more information about the supported ODBC driver managers to configure the Amazon Redshift ODBC drivers, see Using an Amazon Redshift ODBC driver on macOS for macOS operating systems. Also, see "Specifying ODBC Driver Managers on Non- Windows Machines" in the Amazon Redshift ODBC connector installation and configuration guide.

Three files are required for configuring the Amazon Redshift ODBC driver: `amazon.redshiftodbc.ini`, `odbc.ini`, and `odbcinst.ini`.

If you installed to the default location, the `amazon.redshiftodbc.ini` configuration file is located in `/opt/amazon/redshift/lib`.

Additionally, under /opt/amazon/redshift/Setup on macOS, there are sample odbc.ini and odbcinst.ini files. You can use these files as examples for configuring the Amazon Redshift ODBC driver and the data source name (DSN).

We don't recommend using the Amazon Redshift ODBC driver installation directory for the configuration files. The sample files in the Setup directory are for example purposes only. If you reinstall the Amazon Redshift ODBC driver at a later time, or upgrade to a newer version, the installation directory is overwritten. You then lose any changes that you might have made to those files.

To avoid this, copy the amazon.redshiftodbc.ini file to a directory other than the installation directory. If you copy this file to the user's home directory, add a period (.) to the beginning of the file name to make it a hidden file.

For the odbc.ini and odbcinst.ini files, either use the configuration files in the user's home directory or create new versions in another directory. By default, your macOS operating system should have an odbc.ini file and an odbcinst.ini file in the user's home directory (/home/$USER or ~/.). These default files are hidden files, which is indicated by the dot (.) in front of each file name. These files display only when you use the -a flag to list the directory contents.

Whichever option you choose for the odbc.ini and odbcinst.ini files, modify the files to add driver and DSN configuration information. If you create new files, you also need to set environment variables to specify where these configuration files are located.

By default, ODBC driver managers are configured to use hidden versions of the odbc.ini and odbcinst.ini configuration files (named .odbc.ini and .odbcinst.ini) located in the home directory. They also are configured to use the amazon.redshiftodbc.ini file in the /lib subfolder of the driver installation directory. If you store these configuration files elsewhere, set the environment variables described following so that the driver manager can locate the files. For more information, see "Specifying the Locations of the Driver Configuration Files" in the Amazon Redshift ODBC connector installation and configuration guide.

Creating a data source name macOS X operating systems

When connecting to your data store using a data source name (DSN), configure the odbc.ini file to define DSNs. Set the properties in the odbc.ini file to create a DSN that specifies the connection information for your data store.

For information about how to configure the odbc.ini file, see "Creating a Data Source Name on a Non-Windows Machine" in the Amazon Redshift ODBC connector installation and configuration guide.

Use the following format on macOS operating systems.

```
[ODBC Data Sources]
driver_name=dsn_name

[dsn_name]
Driver=path/lib/amazonredshiftodbc.dylib

Host=cluster_endpoint
Port=port_number
Database=database_name
locale=locale
```

The following example shows the configuration for odbc.ini on macOS operating systems.

```
[ODBC Data Sources]
Amazon_Redshift_dylib=Amazon Redshift DSN for macOS

[Amazon Redshift DSN for macOS]
Driver=/opt/amazon/redshift/lib/amazonredshiftodbc.dylib
Host=examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com
Port=5932
Database=dev
locale=en-US
```

Configuring a connection without a DSN on macOS operating systems

To connect to your data store through a connection that doesn't have a DSN, define the driver in the odbcinst.ini file. Then provide a DSN-less connection string in your application.

For information about how to configure the odbcinst.ini file in this case, see "Configuring a DSN-less Connection on a Non-Windows Machine" in the Amazon Redshift ODBC connector installation and configuration guide.

Use the following format on macOS operating systems.

```
[ODBC Drivers]
driver_name=Installed
```

```
...

[driver_name]
Description=driver_description
Driver=path/lib/amazonredshiftodbc.dylib

...
```

The following example shows the `odbcinst.ini` configuration for the driver installed in the default directory on macOS operating systems.

```
[ODBC Drivers]
Amazon RedshiftODBC DSN=Installed

[Amazon RedshiftODBC DSN]
Description=Amazon Redshift ODBC Driver for macOS
Driver=/opt/amazon/redshift/lib/amazonredshiftodbc.dylib
```

Configuring environment variables

Use the correct ODBC driver manager to load the correct driver. To do this, set the library path environment variable. For more information, see "Specifying ODBC Driver Managers on Non-Windows Machines" in the Amazon Redshift ODBC connector installation and configuration guide.

By default, ODBC driver managers are configured to use hidden versions of the `odbc.ini` and `odbcinst.ini` configuration files (named .`odbc.ini` and .`odbcinst.ini`) located in the home directory. They also are configured to use the `amazon.redshiftodbc.ini` file in the `/lib` subfolder of the driver installation directory. If you store these configuration files elsewhere, the environment variables so that the driver manager can locate the files. For more information, see "Specifying the Locations of the Driver Configuration Files" in *Amazon Redshift ODBC Connector Installation and Configuration Guide*.

Configuring connection features

You can configure the following connection features for your ODBC setting:

- Configure the ODBC driver to provide credentials and authenticate the connection to the Amazon Redshift database.
- Configure the ODBC driver to connect to a socket enabled with Secure Sockets Layer (SSL), if you are connecting to an Amazon Redshift server that has SSL enabled.

- Configure the ODBC driver to connect to Amazon Redshift through a proxy server.
- Configure the ODBC driver to use a query processing mode to prevent queries from consuming too much memory.
- Configure the ODBC driver to pass IAM authentication processes through a proxy server.
- Configure the ODBC driver to use TCP keepalives to prevent connections from timing out.

For information about these connection features, see the Amazon Redshift ODBC connector installation and configuration guide.

ODBC driver options

You can use configuration options to control the behavior of the Amazon Redshift ODBC driver.

In Microsoft Windows, you typically set driver options when you configure a data source name (DSN). You can also set driver options in the connection string when you connect programmatically, or by adding or changing registry keys in `HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI\your_DSN`. For more information about configuring a DSN, see Using an Amazon Redshift ODBC driver on Microsoft Windows.

In macOS, you set driver configuration options in your `odbc.ini` and `amazon.redshiftodbc.ini` files, as described in Use an ODBC driver manager to configure the driver. Configuration options set in an `amazon.redshiftodbc.ini` file apply to all connections. In contrast, configuration options set in an `odbc.ini` file are specific to a connection. Configuration options set in `odbc.ini` take precedence over configuration options set in `amazon.redshiftodbc.ini`.

For information about how to set up ODBC driver configuration options, see the Amazon Redshift ODBC connector installation and configuration guide.

Previous ODBC driver versions

Download a previous version of the Amazon Redshift ODBC driver only if your tool requires a specific version of the driver.

Previous ODBC driver versions for Windows

The following are the 64-bit drivers:

- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.6.3.1008/AmazonRedshiftODBC64-1.6.3.1008.msi

- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.6.3.1006/AmazonRedshiftODBC64-1.6.3.1006.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.6.1.1000/AmazonRedshiftODBC64-1.6.1.1000.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.5.20.1024/AmazonRedshiftODBC64-1.5.20.1024.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.5.16.1019/AmazonRedshiftODBC64-1.5.16.1019.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.5.9.1011/AmazonRedshiftODBC64-1.5.9.1011.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.5.7.1007/AmazonRedshiftODBC64-1.5.7.1007.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.65.1000/AmazonRedshiftODBC64-1.4.65.1000.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.62.1000/AmazonRedshiftODBC64-1.4.62.1000.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.59.1000/AmazonRedshiftODBC64-1.4.59.1000.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.56.1000/AmazonRedshiftODBC64-1.4.56.1000.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.53.1000/AmazonRedshiftODBC64-1.4.53.1000.msi
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.52.1000/AmazonRedshiftODBC64-1.4.52.1000.msi

32-bit drivers are discontinued and previous versions are not supported.

Previous ODBC driver versions for Linux

The following are the versions of the 64-bit driver:

- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.6.3.1008/AmazonRedshiftODBC-64-bit-1.6.3.1008-1.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.6.3.1006/AmazonRedshiftODBC-64-bit-1.6.3.1006-1.x86_64.rpm

- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.6.1.1000/AmazonRedshiftODBC-64-bit-1.6.1.1000-1.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.5.20.1024/AmazonRedshiftODBC-64-bit-1.5.20.1024-1.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.5.16.1019/AmazonRedshiftODBC-64-bit-1.5.16.1019-1.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.5.9.1011/AmazonRedshiftODBC-64-bit-1.5.9.1011-1.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.5.7.1007/AmazonRedshiftODBC-64-bit-1.5.7.1007-1.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.65.1000/AmazonRedshiftODBC-64-bit-1.4.65.1000-1.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.62.1000/AmazonRedshiftODBC-64-bit-1.4.62.1000-1.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.59.1000/AmazonRedshiftODBC-64-bit-1.4.59.1000-1.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.59.1000/AmazonRedshiftODBC-64-bit-1.4.59.1000-1.x86_64.deb
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.56.1000/AmazonRedshiftODBC-64-bit-1.4.56.1000-1.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.56.1000/AmazonRedshiftODBC-64-bit-1.4.56.1000-1.x86_64.deb
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.52.1000/AmazonRedshiftODBC-64-bit-1.4.52.1000-1.x86_64.rpm
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.52.1000/AmazonRedshiftODBC-64-bit-1.4.52.1000-1.x86_64.deb

32-bit drivers are discontinued and previous versions are not supported.

Previous ODBC driver versions for macOS

The following are the versions of the Amazon Redshift ODBC driver for macOS:

- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.6.3.1008/AmazonRedshiftODBC-64-bit.1.6.3.1008.universal.pkg

- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.6.3.1006/AmazonRedshiftODBC-64-bit.1.6.3.1006.universal.pkg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.6.1.1000/AmazonRedshiftODBC-64-bit.1.6.1.1000.universal.pkg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.5.20.1024/AmazonRedshiftODBC-1.5.20.1024.arm64.dmg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.5.20.1024/AmazonRedshiftODBC-1.5.20.1024.x86_64.dmg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.5.16.1019/AmazonRedshiftODBC-1.5.16.1019.x86_64.dmg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.5.9.1011/AmazonRedshiftODBC-1.5.9.1011.x86_64.dmg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.5.7.1007/AmazonRedshiftODBC-1.5.7.1007.x86_64.dmg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.65.1000/AmazonRedshiftODBC-1.4.65.1000.dmg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.62.1000/AmazonRedshiftODBC-1.4.62.1000.dmg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.59.1000/AmazonRedshiftODBC-1.4.59.1000.dmg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.56.1000/AmazonRedshiftODBC-1.4.56.1000.dmg
- https://s3.amazonaws.com/redshift-downloads/drivers/odbc/1.4.52.1000/AmazonRedshiftODBC-1.4.52.1000.dmg

Best practices for discovering metadata

Amazon Redshift provides multiple ways to discover metadata. For applications and tools, including those built with AI agents, we recommend that you use the driver metadata API or `SHOW` commands for consistent, reliable, and performant metadata discovery.

Topics

- Use the Amazon Redshift driver metadata API for applications and tools

- Use SHOW commands
- Querying system tables
- Set the application name connection property

Use the Amazon Redshift driver metadata API for applications and tools

For applications and tools that connect to Amazon Redshift, such as a business intelligence tool or query editor, we recommend that you use the driver metadata API provided by the Amazon Redshift JDBC 2.x, ODBC 2.x, or Python drivers to discover metadata about your data warehouse objects, including databases, schemas, tables, columns, and data types. As alternatives, you can use Amazon Redshift `SHOW` commands.

Use the driver metadata API for the following benefits:

- **Specification-compliant**. The JDBC and ODBC drivers implement standard metadata interfaces (`DatabaseMetaData` in JDBC, `SQLTables` and `SQLColumns` in ODBC). Since Python's DB-API (PEP 249) does not define a metadata API specification, the Amazon Redshift Python driver follows the JDBC DatabaseMetaData specification, providing equivalent methods such as `get_tables()`, `get_columns()`, and `get_schemas()`. These APIs follow well-defined specifications, so your integration code is portable. As Amazon Redshift evolves its internal system tables, your application doesn't need to change.
- **Performance-optimized**. The driver metadata API is optimized to return metadata efficiently. AWS continues to invest in driver metadata API performance.
- **Forward-compatible**. Amazon Redshift adheres to the JDBC, ODBC, and Python connector specifications. When you code against these standard APIs, your application is protected from changes to the underlying system catalog structure.

Example: Using JDBC DatabaseMetaData.getTables() to retrieve table metadata

```
DatabaseMetaData dbmd = connection.getMetaData();

// getTables(catalog, schemaPattern, tableNamePattern, types)
//   catalog:          "test" – filters to the database named "test"
//   schemaPattern:    "test_pattern" – filters schemas matching this pattern (supports
 SQL wildcards % and _)
//   tableNamePattern: null – no filter, returns all table names
//   types:            {"TABLE", "EXTERNAL TABLE"} – only return regular tables and
 external tables
```

```
ResultSet rs = dbmd.getTables("test", "test_pattern", null, new String[] {"TABLE",
 "EXTERNAL TABLE"});
```

Example: Using Python cursor.get_columns() to retrieve column metadata

```
cursor: redshift_connector.Cursor = conn.cursor()

# get_columns(catalog, schema_pattern, table_name_pattern, column_name_pattern)
#   catalog:             'test' – filters to the database named "test"
#   schema_pattern:      'test_pattern' – filters schemas matching this pattern
 (supports SQL wildcards % and _)
#   table_name_pattern:  'testabc' – filters to the table named "testabc"
#   column_name_pattern: '%' – wildcard, returns all columns in the matching table
result: tuple = cursor.get_columns('test', 'test_pattern', 'testabc', '%')
```

Example: Using ODBC SQLPrimaryKeys() to retrieve primary key metadata

```
// SQLPrimaryKeys(hstmt, catalog, catalog_len, schema, schema_len, table, table_len)
//   catalog: "test"         – filters to the database named "test"
//   schema:  "test_schema"  – filters to the schema named "test_schema"
//   table:   "test_table"   – retrieves primary key columns for this table
// Note: Unlike getTables/getColumns, SQLPrimaryKeys does NOT support wildcard
 patterns.
retcode = SQLPrimaryKeys(hstmt, (SQLCHAR *)"test", SQL_NTS, (SQLCHAR *)"test_schema",
 SQL_NTS, (SQLCHAR *)"test_table", SQL_NTS);

while (SQL_SUCCEEDED(retcode = SQLFetch(hstmt))) {
    for (i = 1; i <= columns; i++) {
        retcode = SQLGetData(hstmt, i, SQL_C_CHAR, buf, sizeof(buf), &indicator);
    }
}
```

Example: Using ODBC SQLTables() to list databases and schemas

The ODBC API does not provide separate functions for listing catalogs or schemas. Instead, you use special calling conventions of `SQLTables()` to retrieve this information.

To list all databases (catalogs)

Call `SQLTables()` with `CatalogName` set to `SQL_ALL_CATALOGS`. Set `SchemaName` and `TableName` to empty strings. The result set returns valid values only in the `TABLE_CAT` column. All other columns contain NULLs.

```
// List all catalogs (databases) available on the data source.
retcode = SQLTables(hstmt,
    (SQLCHAR *)SQL_ALL_CATALOGS, SQL_NTS, // CatalogName = "%" (SQL_ALL_CATALOGS)
    (SQLCHAR *)"", 0,                     // SchemaName = "" (empty string)
    (SQLCHAR *)"", 0,                     // TableName  = "" (empty string)
    NULL, 0);                             // TableType  = NULL (not filtered)
```

To list all schemas

Call `SQLTables()` with `SchemaName` set to `SQL_ALL_SCHEMAS`. Set `CatalogName` and `TableName` to empty strings.

```
// List all schemas available on the data source.
retcode = SQLTables(hstmt,
    (SQLCHAR *)"", 0,                     // CatalogName = "" (empty string)
    (SQLCHAR *)SQL_ALL_SCHEMAS, SQL_NTS,  // SchemaName = "%" (SQL_ALL_SCHEMAS)
    (SQLCHAR *)"", 0,                     // TableName  = "" (empty string)
    NULL, 0);                             // TableType  = NULL (not filtered)
```

Note

The ODBC specification defines only `TABLE_SCHEM` as valid for schema enumeration. Amazon Redshift also populates `TABLE_CAT` because it supports cross-database metadata discovery, and each schema is scoped to a specific database.

Use SHOW commands

If the driver metadata API doesn't cover your use case, you can use Amazon Redshift `SHOW` commands to retrieve metadata. `SHOW` commands are optimized for fast metadata retrieval. They're useful when you need metadata in an interactive SQL session, or when your application connects through a client that doesn't expose the driver metadata API.

The following `SHOW` commands are supported for common discovery operations:

- SHOW DATABASES
- SHOW SCHEMAS
- SHOW TABLES
- SHOW COLUMNS

- SHOW CONSTRAINTS
- SHOW GRANTS
- SHOW FUNCTIONS
- SHOW PROCEDURES
- SHOW PARAMETERS

Examples

```
SHOW SCHEMAS FROM DATABASE dev;

database_name |     schema_name      | schema_owner | schema_type |         schema_acl
         | source_database | schema_option
--------------+----------------------+--------------+-------------
+-----------------------------+-----------------+--------------
dev           | pg_automv            |            1 | local       |
         |                 |
dev           | pg_catalog           |            1 | local       | jpuser=UC/
jpuser~=U/jpuser  |                 |
dev           | public               |            1 | local       | jpuser=UC/
jpuser~=UC/jpuser |                 |
dev           | information_schema   |            1 | local       | jpuser=UC/
jpuser~=U/jpuser  |                 |
dev           | schemad79cd6d93bf043 |            1 | local       |
         |                 |
```

```
SHOW TABLES FROM SCHEMA dev.s1 LIKE '%view' LIMIT 1;

database_name | schema_name |    table_name     | table_type |              table_acl
             | remarks | owner | last_altered_time | last_modified_time | dist_style |
  table_subtype
--------------+-------------+-------------------+------------
+--------------------------------------+---------+-------+------------------
+--------------------+------------+-------------------
dev           | s1          | late_binding_view | VIEW       | {alice=arwdRxtDPA/
alice,bob=d/alice} |         | alice |                   |                    |
    | LATE BINDING VIEW
```

```
SHOW COLUMNS FROM TABLE second_db.public.t22;
```

```
 database_name | schema_name | table_name | column_name | ordinal_position |
 column_default | is_nullable |          data_type          | character_maximum_length
 | numeric_precision | numeric_scale | remarks | sort_key_type | sort_key | dist_key |
 encoding | collation
---------------+-------------+------------+-------------+------------------
+----------------+-------------+----------------------------
+-------------------------+------------------+---------------+---------
+---------------+----------+----------+----------+-----------
 second_db     | public      | t22        | col1        |                1 |
     | YES         | integer                     |                          |
      32 |             0 |         | INTERLEAVED   |       -1 |          | mostly8  |
 second_db     | public      | t22        | col2        |                2 |
      | YES         | character varying           |                      100 |
           |               |         | INTERLEAVED   |        2 |          | text255  |
 default
 second_db     | public      | t22        | col3        |                3 |
     | YES         | timestamp without time zone |                          |
         |               |         |               |        0 |          | raw      |
 second_db     | public      | t22        | col4        |                4 |
     | YES         | numeric                     |                          |
      10 |             2 |         |               |        0 |          | az64     |
```

For more information, see SHOW in the *Amazon Redshift Database Developer Guide*.

Querying system tables

If neither the driver metadata API nor `SHOW` commands meet your needs, you can query PostgreSQL catalog tables (`pg_*`), the information schema (`information_schema`), or Amazon Redshift system views such as `SVV_TABLE_INFO` directly.

This approach is useful for ad hoc exploration, debugging, or administrative tasks that you run from interactive SQL queries, for example in Amazon Redshift query editor v2 or `psql`. However, system catalog structures can change across Amazon Redshift releases. We don't recommend that you rely on direct system table queries as the metadata interface for production application integrations.

When you query system tables, follow these practices:

- **Specify database, schema, and table**. For best performance, include the database name, schema name, and table name in your query filters. Narrowing the scope reduces the amount of metadata that Amazon Redshift needs to scan and return.

- **Use equality or LIKE filters**. Use equality (=) or `LIKE` predicates in your `WHERE` clause to narrow results. Avoid unfiltered queries against large catalog views. They can be slow on clusters with many objects.
- **Avoid functions in filter predicates**. Don't use functions such as `LOWER( )` or string concatenation in your `WHERE` clause when you filter system table queries. These prevent efficient metadata lookups and can significantly degrade query performance.

Example Anti-pattern vs. Recommended Queries

```
-- Avoid:
SELECT * FROM svv_all_columns;

-- Recommended:
SELECT * FROM svv_all_columns WHERE database_name = 'dev' and schema_name = 'public'
 AND table_name = 'sales';
```

```
-- Avoid:
SELECT * FROM svv_all_columns WHERE LOWER(schema_name) = 'public';

-- Recommended:
SELECT * FROM svv_all_columns WHERE schema_name = 'public' AND table_name LIKE 'sales
%';
```

```
-- Avoid:
SELECT * FROM svv_all_columns WHERE database_name || '.' || schema_name || '.' ||
 table_name = 'dev.public.sales';

-- Recommended:
SELECT * FROM svv_all_columns WHERE database_name = 'dev' and schema_name = 'public'
 AND table_name = 'sales';
```

```
-- Avoid:
SELECT * FROM svv_all_columns WHERE CONCAT(database_name, '.', schema_name) =
 'dev.public';

-- Recommended:
SELECT * FROM svv_all_columns WHERE database_name = 'dev' and schema_name = 'public';
```

Set the application name connection property

When you connect to Amazon Redshift, set the optional application name driver property. This property identifies which application or tool established each connection, which makes it easier to monitor, audit, and troubleshoot your Amazon Redshift environment.

The application name that you provide appears in the `application_name` column of `SYS_CONNECTION_LOG`. You can use it to do the following:

- **Track connection sources**. Identify which applications, services, or tools connect to your cluster.
- **Troubleshoot performance**. Correlate queries with the originating application to isolate performance issues.
- **Audit access**. Understand application-level usage patterns across your data warehouse.

The following table shows how to set the application name for each Amazon Redshift driver.

Driver	Property	Example
JDBC	`ApplicationName`	`jdbc:redshift:// endpoint:port/databas ApplicationName=my-etl-pipeline`
ODBC	`ApplicationName`	Set `ApplicationName=my-bi-tool` in the DSN or connection string.
Python	`application_name`	`redshift_connector.connect(..., application_name='my-data-catalog')`

For more information about the application name property for each driver, see the following:

- JDBC: ApplicationName
- ODBC: ApplicationName
- Python: application_name

Note

Use a descriptive, consistent naming convention for application names, such as *team*-*service*-*environment* (for example, `quicksight-analytics-dashboard-prod`). This approach makes it easier to filter and analyze `SYS_CONNECTION_LOG` data.

Configuring security options for connections

Amazon Redshift supports Secure Sockets Layer (SSL) connections to encrypt data and server certificates to validate the server certificate that the client connects to.

SSL

To support SSL connections, Amazon Redshift creates and installs an AWS Certificate Manager (ACM) issued SSL certificate on each cluster. ACM certificates are publicly trusted by most operating systems, web browsers, and clients. You might need to download a certificate bundle if your SQL clients or applications connect to Amazon Redshift using SSL with the `sslmode` connection option set to `require`, `verify-ca`, or `verify-full`. If your client needs a certificate, Amazon Redshift provides a bundle certificate as follows:

Note

We recommend using `verify-full` for SSL connections whenever possible.

- Download the bundle from https://s3.amazonaws.com/redshift-downloads/amazon-trust-ca-bundle.crt.
  - The expected MD5 checksum number is 418dea9b6d5d5de7a8f1ac42e164cdcf.
  - The sha256 checksum number is 36dba8e4b8041cd14b9d60158893963301bcbb92e1c456847784de2acb5bd550.

  Don't use the previous certificate bundle that was located at `https://s3.amazonaws.com/redshift-downloads/redshift-ca-bundle.crt`.
- In the China AWS Region, download the bundle from https://s3.cn-north-1.amazonaws.com.cn/redshift-downloads-cn/amazon-trust-ca-bundle.crt.
  - The expected MD5 checksum number is 418dea9b6d5d5de7a8f1ac42e164cdcf.

- The sha256 checksum number is 36dba8e4b8041cd14b9d60158893963301bcbb92e1c456847784de2acb5bd550.

Don't use the previous certificate bundles that were located at `https://s3.cn-north-1.amazonaws.com.cn/redshift-downloads-cn/redshift-ca-bundle.crt` and `https://s3.cn-north-1.amazonaws.com.cn/redshift-downloads-cn/redshift-ssl-ca-cert.pem`

> **Important**
>
> Amazon Redshift has changed the way that SSL certificates are managed. You might need to update your current trust root CA certificates to continue to connect to your clusters using SSL. For more information, see Transitioning to ACM certificates for SSL connections.

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.

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.

> **Important**
>
> Enable FIPS-compliant SSL mode only if your system is required to be FIPS-compliant.

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 Amazon Redshift cluster or Redshift Serverless workgroup. For information about modifying a parameter group on a cluster, see Amazon Redshift parameter groups. For information about modifying a parameter group on a workgroup, see Configuring a FIPS-compliant SSL connection to Amazon Redshift Serverless.

Amazon Redshift supports the Elliptic Curve Diffie—Hellman Ephemeral (ECDHE) key agreement protocol. With ECDHE, the client and server each have an elliptic curve public-private key pair that is used to establish a shared secret over an insecure channel. You don't need to configure anything in Amazon Redshift to enable ECDHE. If you connect from a SQL client tool that uses ECDHE to

encrypt communication between the client and server, Amazon Redshift uses the provided cipher list to make the appropriate connection. For more information, see Elliptic curve diffie—hellman on Wikipedia and Ciphers on the OpenSSL website.

SSL and trust CA certificates in ODBC

If you connect using the latest Amazon Redshift ODBC drivers (version 1.3.7.1000 or later), you can skip this section. To download the latest drivers, see Configuring a connection for ODBC driver version 2.x for Amazon Redshift.

You might need to update your current trust root CA certificates to continue to connect to your clusters using SSL. For more information, see SSL.

You can verify that the certificate that you downloaded matches the expected MD5 checksum number. To do this, you can use the Md5sum program on Linux operating systems, or another tool on Windows and macOS operating systems.

ODBC DSNs contain an `sslmode` setting that determines how to handle encryption for client connections and server certificate verification. Amazon Redshift supports the following `sslmode` values from the client connection:

- `disable`

  SSL is disabled and the connection is not encrypted.
- `allow`

  SSL is used if the server requires it.
- `prefer`

  SSL is used if the server supports it. Amazon Redshift supports SSL, so SSL is used when you set `sslmode` to `prefer`.
- `require`

  SSL is required.
- `verify-ca`

  SSL must be used and the server certificate must be verified.
- `verify-full`

SSL must be used. The server certificate must be verified and the server hostname must match the hostname attribute on the certificate.

You can determine whether SSL is used and server certificates are verified in a connection between the client and the server. To do this, you need to review the `sslmode` setting for your ODBC DSN on the client and the `require_SSL` setting for the Amazon Redshift cluster on the server. The following table describes the encryption result for the various client and server setting combinations:

sslmode (client)	require_SSL (server)	Result
`disable`	`false`	The connection is not encrypted.
`disable`	`true`	The connection can't be made because the server requires SSL and the client has SSL disabled for the connection.
`allow`	`true`	The connection is encrypted.
`allow`	`false`	The connection is not encrypted.
`prefer` or `require`	`true`	The connection is encrypted.
`prefer` or `require`	`false`	The connection is encrypted.
`verify-ca`	`true`	The connection is encrypted and the server certificate is verified.
`verify-ca`	`false`	The connection is encrypted and the server certificate is verified.
`verify-full`	`true`	The connection is encrypted and the server certificate and hostname are verified.
`verify-full`	`false`	The connection is encrypted and the server certificate and hostname are verified.

Connect using the server certificate with ODBC on Microsoft Windows

If you want to connect to your cluster using SSL and the server certificate, first download the certificate to your client computer or Amazon EC2 instance. Then configure the ODBC DSN.

1. Download the Amazon Redshift certificate authority bundle to your client computer at the `lib` folder in your driver installation directory, and save the file as `root.crt`. For download information, see SSL.
2. Open **ODBC Data Source Administrator**, and add or edit the system DSN entry for your ODBC connection. For **SSL Mode**, select `verify-full` unless you use a DNS alias. If you use a DNS alias, select `verify-ca`. Then choose **Save**.

   For more information about configuring the ODBC DSN, see Configuring a connection for ODBC driver version 2.x for Amazon Redshift.

SSL and server certificates in Java

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. For more information about using server certificates with JDBC, go to Configuring the client in the PostgreSQL documentation.

Connect using trust CA certificates in Java

> **Important**
>
> Amazon Redshift has changed the way that SSL certificates are managed. You might need to update your current trust root CA certificates to continue to connect to your clusters using SSL. For more information, see SSL.

To connect using trust CA certificates

You can use the `redshift-keytool.jar` file to import CA certificates in the Amazon Redshift Certificate Authority bundle into a Java TrustStore or your private TrustStore.

1. If you use the Java command line `-Djavax.net.ssl.trustStore` option, remove it from command line, if possible.

2. Download redshift-keytool.jar.
3. Do one of the following:

   - To import the Amazon Redshift Certificate Authority bundle into a Java TrustStore, run the following command.

     ```
     java -jar redshift-keytool.jar -s
     ```

   - To import the Amazon Redshift Certificate Authority bundle into your private TrustStore, run the following command:

     ```
     java -jar redshift-keytool.jar -k <your_private_trust_store> -
     p <keystore_password>
     ```

Transitioning to ACM certificates for SSL connections

Amazon Redshift is replacing the SSL certificates on your clusters with AWS Certificate Manager (ACM) issued certificates. ACM is a trusted public certificate authority (CA) that is trusted by most current systems. You might need to update your current trust root CA certificates to continue to connect to your clusters using SSL.

This change affects you only if all of the following apply:

- Your SQL clients or applications connect to Amazon Redshift clusters using SSL with the `sslMode` connection option set to `require`, `verify-ca`, or `verify-full` configuration option.
- You aren't using the Amazon Redshift ODBC or JDBC drivers, or you use Amazon Redshift drivers before ODBC version 1.3.7.1000 or JDBC version 1.2.8.1005.

If this change affects you on commercial Amazon Redshift Regions, then you must update your current trust root CA certificates before October 23, 2017. Amazon Redshift will transition your clusters to use ACM certificates between now and October 23, 2017. The change should have very little or no effect on your cluster's performance or availability.

If this change affects you on AWS GovCloud (US) (US) Regions, then you must update your current trust root CA certificates before April 1, 2020 to avoid service interruption. Beginning on this date, clients connecting to Amazon Redshift clusters using SSL encrypted connections need an additional trusted certificate authority (CA). Clients use trusted certificate authorities to confirm

the identity of the Amazon Redshift cluster when they connect to it. Your action is required to update your SQL clients and applications to use an updated certificate bundle that includes the new trusted CA.

> **Important**
>
> In the China Regions on January 5, 2021, Amazon Redshift is replacing the SSL certificates on your clusters with AWS Certificate Manager (ACM) issued certificates. If this change affects you on China (Beijing) Region or China (Ningxia) Region, then you must update your current trust root CA certificates before January 5, 2021 to avoid service interruption. Beginning on this date, clients connecting to Amazon Redshift clusters using SSL encrypted connections need an additional trusted certificate authority (CA). Clients use trusted certificate authorities to confirm the identity of the Amazon Redshift cluster when they connect to it. Your action is required to update your SQL clients and applications to use an updated certificate bundle that includes the new trusted CA.

- Using the latest Amazon Redshift ODBC or JDBC drivers
- Using earlier Amazon Redshift ODBC or JDBC drivers
- Using other SSL connection types

Using the latest Amazon Redshift ODBC or JDBC drivers

The preferred method is to use the latest Amazon Redshift ODBC or JDBC drivers. Amazon Redshift drivers beginning with ODBC version 1.3.7.1000 and JDBC version 1.2.8.1005 automatically manage the transition from an Amazon Redshift self-signed certificate to an ACM certificate. To download the latest drivers, see Configuring a connection for JDBC driver version 2.x for Amazon Redshift.

If you use the latest Amazon Redshift JDBC driver, it's best not to use `-Djavax.net.ssl.trustStore` in JVM options. If you must use `-Djavax.net.ssl.trustStore`, import the Redshift certificate authority bundle into the truststore it points to. For download information, see SSL. For more information, see Importing the Amazon Redshift certificate authority bundle into a TrustStore.

Using earlier Amazon Redshift ODBC or JDBC drivers

- If your ODBC DSN is configured with `SSLCertPath`, overwrite the certificate file in the specified path.
- If `SSLCertPath` is not set, then overwrite the certificate file named `root.crt` in the driver DLL location.

If you must use an Amazon Redshift JDBC driver before version 1.2.8.1005, then do one of the following:

- If your JDBC connection string uses the `sslCert` option, remove the `sslCert` option. Then import the Redshift certificate authority bundle to your Java TrustStore. For download information, see SSL. For more information, see Importing the Amazon Redshift certificate authority bundle into a TrustStore.
- If you use the Java command line `-Djavax.net.ssl.trustStore` option, remove it from command line, if possible. Then import the Redshift certificate authority bundle to your Java TrustStore. For download information, see SSL. For more information, see Importing the Amazon Redshift certificate authority bundle into a TrustStore.

Importing the Amazon Redshift certificate authority bundle into a TrustStore

You can use `redshift-keytool.jar` to import CA certificates in the Amazon Redshift Certificate Authority bundle into a Java TrustStore or your private truststore.

To import the Amazon Redshift certificate authority bundle into a TrustStore

1. Download redshift-keytool.jar.
2. Do one of the following:
   - To import the Amazon Redshift Certificate Authority bundle into a Java TrustStore, run the following command.

     ```
     java -jar redshift-keytool.jar -s
     ```

   - To import the Amazon Redshift Certificate Authority bundle into your private TrustStore, run the following command:

```
java -jar redshift-keytool.jar -k <your_private_trust_store> -
p <keystore_password>
```

Using other SSL connection types

Follow the steps in this section if you connect using any of the following:

- Open source ODBC driver
- Open source JDBC driver
- The Amazon Redshift RSQL command line interface
- Any language bindings based on libpq, such as psycopg2 (Python) and ruby-pg (Ruby)

To use ACM certificates with other SSL connection types:

1. Download the Amazon Redshift certificate authority bundle. For download information, see SSL.
2. Place the certificates from the bundle in your `root.crt` file.
   - On Linux and macOS operating systems, the file is `~/.postgresql/root.crt`.
   - On Microsoft Windows, the file is `%APPDATA%\postgresql\root.crt`.

Connecting from client tools and code

Amazon Redshift provides Amazon Redshift query editor v2 to connect to your clusters and workgroups. For more information, see Querying a database using the query editor v2.

This section provides some options for third-party tools to connect. Additionally, it describes how to connect to your cluster programmatically.

Topics

- Connecting with Amazon Redshift RSQL
- Connect to a cluster with Amazon Redshift RSQL
- Amazon Redshift RSQL meta commands
- Amazon Redshift RSQL variables

- Amazon Redshift RSQL error codes
- Amazon Redshift RSQL environment variables

Connecting with Amazon Redshift RSQL

Amazon Redshift RSQL is a command-line client for interacting with Amazon Redshift clusters and databases. You can connect to an Amazon Redshift cluster, describe database objects, query data, and view query results in various output formats.

Amazon Redshift RSQL supports the capabilities of the PostgreSQL psql command-line tool with an additional set of capabilities specific to Amazon Redshift. These include the following:

- You can use single sign-on authentication using AD FS, PingIdentity, Okta, Azure ADm or other SAML/JWT based identity providers. You can also use browser-based SAML identity providers for multi-factor authentication (MFA).
- You can describe properties or attributes of Amazon Redshift objects such as table distribution keys, table sort keys, late-binding views (LBVs), and materialized views. You can also describe properties or attributes of external tables in an AWS Glue catalog or Apache Hive Metastore, external databases in Amazon RDS for PostgreSQL, Amazon Aurora PostgreSQL-Compatible Edition, RDS for MySQL (preview) and Amazon Aurora MySQL-Compatible Edition (preview), and tables shared by using Amazon Redshift data sharing.
- You can also use enhanced control flow commands such as `IF` (`\ELSEIF`, `\ELSE`, `\ENDIF`), `\GOTO` and `\LABEL`.

With Amazon Redshift RSQL batch mode, which runs a script passed as an input parameter, you can run scripts that include both SQL and complex business logic. If you have existing self-managed, on-premises data warehouses, you can use Amazon Redshift RSQL to replace existing extract, transform, load (ETL) and automation scripts, such as Teradata BTEQ scripts. Using RSQL helps you to avoid manually reimplementing scripts in a procedural language.

Amazon Redshift RSQL is available for Linux, Windows, and macOS operating systems.

To report issues for Amazon Redshift RSQL, write to `<redshift-rsql-support@amazon.com>`.

Topics

- Getting started with Amazon Redshift RSQL
- Amazon Redshift RSQL change log

Getting started with Amazon Redshift RSQL

Install Amazon Redshift RSQL on a computer with a Linux, macOS, or Microsoft Windows operating system.

Download RSQL

- Linux 64-bit RPM: RSQL Version 1.1.2
  - Linux Artifact Signature Key: Key
  - Linux Artifact Signed Hash: Hash
- Mac OS 64-bit PKG: RSQL Version 1.1.2
- Windows 64-bit MSI: RSQL Version 1.1.2

See the change log and downloads for previous versions at Amazon Redshift RSQL change log.

Install RSQL for Linux

Follow the steps below to install RSQL for Linux.

1. Install the driver manager with the following command:

```
sudo yum install unixODBC
```

2. Install the ODBC driver: Downloading and installing the Amazon Redshift ODBC driver.
3. Copy the ini file to your home directory:

```
cp /opt/amazon/redshiftodbcx64/odbc.ini ~/.odbc.ini
```

4. Set the environment variables to point to the location of the file:

```
export ODBCINI=~/.odbc.ini
export ODBCSYSINI=/opt/amazon/redshiftodbcx64/
export AMAZONREDSHIFTODBCINI=/opt/amazon/redshiftodbcx64/amazon.redshiftodbc.ini
```

5. You can now install RSQL by running the following command.

```
sudo rpm -i AmazonRedshiftRsql-<version>.rhel.x86_64.rpm
```

Install RSQL for Mac

Follow the steps below to install RSQL for Mac OSX.

1. Install the driver manager with the following command:

```
brew install unixodbc --build-from-source
```

2. Install the ODBC driver: Downloading and installing the Amazon Redshift ODBC driver.
3. Copy the ini file to your home directory:

```
cp /opt/amazon/redshift/Setup/odbc.ini ~/.odbc.ini
```

4. Set the environment variables to point to the location of the file:

```
export ODBCINI=~/.odbc.ini
export ODBCSYSINI=/opt/amazon/redshift/Setup
export AMAZONREDSHIFTODBCINI=/opt/amazon/redshift/lib/amazon.redshiftodbc.ini
```

5. Set `DYLD_LIBRARY_PATH` to location of your libodbc.dylib if its not in `/usr/local/lib`.

```
export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:/usr/local/lib
```

6. Double-click the pkg file to run the installer.
7. Follow the steps in the installer to complete the installation. Agree to the terms of the license agreement.

Install RSQL for Windows

Follow the steps below to install RSQL for Windows.

1. Install the ODBC driver: Downloading and installing the Amazon Redshift ODBC driver.
2. Double-click the RSQL download file to run the installer, then follow the prompts to complete the installation.

Amazon Redshift RSQL change log

1.1.2 (2025-12-11)

Bug Fixes

- Fixed a bug that caused failures with \goto and \label commands.
- Fixed a bug that prevented RSQL from printing variable values when variables were enclosed in quotes.
- Fixed a bug that caused RSQL crashes when query results exceeded the ODBC DSN Fetch parameter size while UseDeclareFetch was enabled.
- Fixed an issue where multiple pages of results are returned at once even though pager is turned on.
- Fixed a bug that triggered RSQL crashes when queries failed within transaction blocks.

1.1.1 (2025-11-20)

Bug Fixes

- Fixed an issue where RSQL incorrectly parsed queries when using the -c flag. This fix applies to all platforms.
- Fix a bug on Mac that prevented users from using the \s command in RSQL.

1.1.0 (2025-11-11)

Bug Fixes

- Resolved a memory leak issue that was causing unexpected crashes in rSQL.
- Removed OpenSSL dependency from RSQL.
- Fixed linking conflicts with libpq/psql installations in the same environment.
- Improved platform compatibility for Amazon Linux 2023, Windows, and macOS.
- Fixed an issue where output would be truncated when exceeding the current display size.

1.0.8 (2023-06-19)

Bug Fixes

- Fixed an issue where output is truncated with SHOW commands.
- Added support to \de for describing external Kinesis streams and Kafka topics.

1.0.7 (2023-03-22)

Bug Fixes

- Fixed an issue where RSQL could not describe materialized views.
- Fixed permission-denied error on stl_connection_log when using Amazon Redshift Serverless.
- Fixed issue where RSQL may process \GOTO labels incorrectly.
- Fixed issue where SSL messages are printed in quiet mode.
- Fixed issue with random characters displayed when describing stored procedures.
- Fixed issue with printing duplicate ERROR/INFO messages.

New

- RSQL now gets SSL information directly from the ODBC driver.

1.0.6 (2023-02-21)

Bug Fixes

- Fixed an issue where \d throws an error - invalid input syntax for integer: "xid" - on Redshift patch 1.0.46086 (P173).

New

- Renamed installation files to reflect supported architecture.

1.0.5 (2022-06-27)

Bug Fixes

- Send SQL error messages to standard error (stderr).
- Fixed issue with exit codes when using ON_ERROR_STOP. Scripts now end after encountering an error and return the correct exit codes.
- Maxerror is now case insensitive.

New

- Added support for ODBC 2.x driver.

1.0.4 (2022-03-19)

- Add support for RSPASSWORD environment variable. Set a password to connect to Amazon Redshift. For example, `export RSPASSWORD=TestPassw0rd`.

1.0.3 (2021-12-08)

Bug Fixes

- Fixed dialogue pop up when using `\c` or `\logon` to switch between databases in Windows OS.
- Fixed crash when checking ssl information.

Amazon Redshift RSQL previous versions

Choose one of the links to download the version of Amazon Redshift RSQL you need, based on your operating system.

Linux 64-bit RPM

- RSQL Version 1.1.1
- RSQL Version 1.1.0
- RSQL Version 1.0.8
- RSQL Version 1.0.7
- RSQL Version 1.0.6
- RSQL Version 1.0.5
- RSQL Version 1.0.4
- RSQL Version 1.0.3
- RSQL Version 1.0.1

Mac OS 64-bit DMG/PKG

- RSQL Version 1.1.1
- RSQL Version 1.1.0
- RSQL Version 1.0.8
- RSQL Version 1.0.7
- RSQL Version 1.0.6
- RSQL Version 1.0.5
- RSQL Version 1.0.4
- RSQL Version 1.0.3
- RSQL Version 1.0.1

Windows 64-bit MSI

- RSQL Version 1.1.1
- RSQL Version 1.1.0
- RSQL Version 1.0.8
- RSQL Version 1.0.7
- RSQL Version 1.0.6
- RSQL Version 1.0.5
- RSQL Version 1.0.4
- RSQL Version 1.0.3
- RSQL Version 1.0.1

Connect to a cluster with Amazon Redshift RSQL

With Amazon Redshift, you can connect to a cluster and interact with it using RSQL. This is a command-line tool that provides a secure way to query data, create database objects, and manage your Amazon Redshift cluster. The following sections guide you through the steps to establish a connection to your cluster using RSQL with and without a data source name (DSN).

Connecting without a DSN

1. On the Amazon Redshift console, choose the cluster you want to connect to and note the endpoint, database, and port.

2. At a command prompt, specify the connection information by using command-line parameters.

```
rsql -h <endpoint> -U <username> -d <databasename> -p <port>
```

Here, the following apply:

- *<endpoint>* is the **Endpoint** you recorded in the previous step.
- *<username>* is the name of a user with permissions to connect to the cluster.
- *<databasename>* is the **Database Name** you recorded in the previous step.
- *<port>* is the **Port** you recorded in the previous step. *<port>* is an optional parameter.

An example follows.

```
rsql -h testcluster.example.amazonaws.com -U user1 -d dev -p 5439
```

3. At the password prompt, enter the password for the *<username>* user.

A successful connection response looks like the following.

```
% rsql -h testcluster.example.com -d dev -U user1 -p 5349
Password for user user1:
DSN-less Connected
DBMS Name: Amazon Redshift
Driver Name: Amazon Redshift ODBC Driver
Driver Version: 1.4.27.1000
Rsql Version: 1.0.1
Redshift Version: 1.0.29306
Type "help" for help.

(testcluster) user1@dev=#
```

The command to connect has the same parameters on Linux, Mac OS, and Windows.

Connecting with a DSN

You can connect RSQL to Amazon Redshift by using a DSN to simplify the organization of connection properties.This topic includes instructions for ODBC-driver installation and descriptions for DSN properties.

Using a DSN connection with a password

The following shows an example of a DSN-connection configuration that uses a password. The default <path to driver> for Mac OSX is /opt/amazon/redshift/lib/libamazonredshiftodbc.dylib and for Linux is /opt/amazon/redshiftodbc/lib/64/libamazonredshiftodbc64.so.

```
[testuser]
Driver=/opt/amazon/redshiftodbc/lib/64/libamazonredshiftodbc64.so
SSLMode=verify-ca
Min_TLS=1.2
boolsaschar=0
Host=<server endpoint>
Port=<database port>
Database=<dbname>
UID=<username>
PWD=<password>
sslmode=prefer
```

The following output results from a successful connection.

```
% rsql -D testuser
DSN Connected
DBMS Name: Amazon Redshift
Driver Name: Amazon Redshift ODBC Driver
Driver Version: 1.4.27.1000
Rsql Version: 1.0.1
Redshift Version: 1.0.29306
Type "help" for help.

(testcluster) user1@dev=#
```

Using Single sign-on DSN

You can configure a DSN for single sign-on authentication. The following shows an example of a DSN-connection configuration that uses Okta single sign on.

```
[testokta]
Driver=<path to driver>
SSLMode=verify-ca
Min_TLS=1.2
```

```
boolsaschar=0
Host=<server endpoint>
clusterid=<cluster id>
region=<region name>
Database=<dbname>
locale=en-US
iam=1
plugin_name=<plugin name>
uid=<okta username>
pwd=<okta password>
idp_host=<idp endpoint>
app_id=<app id>
app_name=<app name>
preferred_role=<role arn>
```

Sample output from a successful connection.

```
% rsql -D testokta
DSN Connected
DBMS Name: Amazon Redshift
Driver Name: Amazon Redshift ODBC Driver
Driver Version: 1.4.27.1000
Rsql Version: 1.0.1
Redshift Version: 1.0.29306
Type "help" for help.

(testcluster) user1@dev=#
```

The following example shows an example of a DSN-connection configuration that uses Azure single sign on.

```
[testazure]
Driver=<path to driver>
SSLMode=verify-ca
Min_TLS=1.2
boolsaschar=0
Host=<server endpoint>
Port=<cluster port>
clusterid=<cluster id>
region=<region name>
Database=<dbname>
locale=en-us
iam=1
```

```
plugin_name=<plugin name>
uid=<azure username>
pwd=<azure password>
idp_tenant=<Azure idp tenant uuid>
client_id=<Azure idp client uuid>
client_secret=<Azure idp client secret>
```

Using a DSN connection with an IAM profile

You can connect to Amazon Redshift using your configured IAM profile. The IAM profile must have privileges to call `GetClusterCredentials`. The following example shows the DSN properties to use. The `ClusterID` and `Region` parameters are required only if the `Host` is not an Amazon provided endpoint like `examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com`.

```
[testiam]
Driver=Default
Host=testcluster.example.com
Database=dev
DbUser=testuser
ClusterID=rsqltestcluster
Region=us-east-1
IAM=1
Profile=default
```

The value for the `Profile` key is the named profile you choose from your AWS CLI credentials. This example shows the credentials for the profile named `default`.

```
$ cat .aws/credentials
[default]
aws_access_key_id = ASIAIOSFODNN7EXAMPLE
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

The following shows the connection response.

```
$ rsql -D testiam
DSN Connected
DBMS Name: Amazon Redshift
Driver Name: Amazon Redshift ODBC Driver
Driver Version: 1.4.27.1000
Rsql Version: 1.0.1
```

```
Redshift Version: 1.0.29306
Type "help" for help.

(testcluster) testuser@dev=>
```

Using a DSN connection with an Instance profile

You can connect to Amazon Redshift using your Amazon EC2 instance profile. The instance profile must have privileges to call `GetClusterCredentials`. See example below for the DSN properties to use. The `ClusterID` and `Region` parameters are required only if the `Host` is not an Amazon provided endpoint like `examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com`.

```
[testinstanceprofile]
Driver=Default
Host=testcluster.example.com
Database=dev
DbUser=testuser
ClusterID=rsqltestcluster
Region=us-east-1
IAM=1
Instanceprofile=1
```

The following shows the connection response.

```
$ rsql -D testinstanceprofile
DSN Connected
DBMS Name: Amazon Redshift
Driver Name: Amazon Redshift ODBC Driver
Driver Version: 1.4.27.1000
Rsql Version: 1.0.1
Redshift Version: 1.0.29306
Type "help" for help.

(testcluster) testuser@dev=>
```

Using a DSN connection with the default credential provider chain

To connect using the default credential provider chain, specify only the IAM property, and Amazon Redshift RSQL will attempt to acquire credentials in the order described in Working with AWS Credentials in the AWS SDK for Java. At least one of the providers in the chain must have

`GetClusterCredentials` permission. This is useful for connecting from ECS containers, for example.

```
[iamcredentials]
Driver=Default
Host=testcluster.example.com
Database=dev
DbUser=testuser
ClusterID=rsqltestcluster
Region=us-east-1
IAM=1
```

Amazon Redshift RSQL meta commands

Amazon Redshift RSQL meta commands return informational records about databases or specific database objects. Results can include various columns and metadata. Other commands perform specific actions. These commands are preceeded with a backslash.

\d[S+]

Lists local user created tables, regular views, late-binding views and materialized views. \dS also lists tables and views, like \d, but system objects are included in the returned records. The + results in the additional metadata column `description` for all listed objects. The following shows sample records returned as a result of the command.

```
List of relations
 schema |   name    | type  |  owner
--------+-----------+-------+---------
 public | category  | table | awsuser
 public | date      | table | awsuser
 public | event     | table | awsuser
 public | listing   | table | awsuser
 public | sales     | table | awsuser
 public | users     | table | awsuser
 public | venue     | table | awsuser
(7 rows)
```

\d[S+] NAME

Describes a table, view, or index. Includes the column names and types. It also provides the diststyle, backup configuration, create date (tables created after October 2018), and constraints.

For example, `\dS+ sample` returns object properties. Appending S+ results in additional columns included in the returned records.

```
Table "public.sample"
 Column |            Type             |   Collation    | Nullable | Default Value |
 Encoding  | DistKey | SortKey
--------+-----------------------------+----------------+----------+---------------
+-----------+---------+---------
 col1   | smallint                    |                | NO       |               |
 none      | t       | 1
 col2   | character(100)              | case_sensitive | YES      |               |
 none      | f       | 2
 col3   | character varying(100)      | case_sensitive | YES      |               |
 text32k   | f       | 3
 col4   | timestamp without time zone |                | YES      |               |
 runlength | f       | 0
 col5   | super                       |                | YES      |               |
 zstd      | f       | 0
 col6   | bigint                      |                | YES      |               |
 az64      | f       | 0

Diststyle: KEY
Backup: YES
Created: 2021-07-20 19:47:27.997045
Unique Constraints:
    "sample_pkey" PRIMARY KEY (col1)
    "sample_col2_key" UNIQUE (col2)
Foreign-key constraints:
    "sample_col2_fkey" FOREIGN KEY (col2) REFERENCES lineitem(l_orderkey)
```

The distribution style, or *Diststyle*, of the table can be KEY, AUTO, EVEN or ALL.

Backup indicates if the table is backed up when a snapshot is taken. Valid values are YES or NO.

Created is the timestamp for when the table is created. The creation date isn't available for Amazon Redshift tables created before November 2018. Tables created before this date display n/a (Not Available).

Unique Constraints lists unique and primary key constraints on the table.

Foreign-key constraints lists foreign-key constraints on the table.

\dC[+] [PATTERN]

Lists casts. Includes the source type, target type, and whether the cast is implicit.

The following shows a subset of results from \dC+.

```
List of casts
         source type         |         target type         |      function       |
 implicit?   | description
-----------------------------+-----------------------------+---------------------
+---------------+-------------
 "char"                      | character                   | bpchar              | in
 assignment |
 "char"                      | character varying           | text                | in
 assignment |
 "char"                      | integer                     | int4                | no
          |
 "char"                      | text                        | text                | yes
          |
 "path"                      | point                       | point               | no
          |
 "path"                      | polygon                     | polygon             | in
 assignment |
 abstime                     | date                        | date                | in
 assignment |
 abstime                     | integer                     | (binary coercible)  | no
          |
 abstime                     | time without time zone      | time                | in
 assignment |
 abstime                     | timestamp with time zone    | timestamptz         | yes
          |
 abstime                     | timestamp without time zone | timestamp           | yes
          |
 bigint                      | bit                         | bit                 | no
          |
 bigint                      | boolean                     | bool                | yes
          |
 bigint                      | character                   | bpchar              | in
 assignment |
 bigint                      | character varying           | text                | in
 assignment |
 bigint                      | double precision            | float8              | yes
          |
```

```
bigint                      | integer                     | int4               | in
assignment |
bigint                      | numeric                     | numeric            | yes
         |
bigint                      | oid                         | oid                | yes
         |
bigint                      | real                        | float4             | yes
         |
bigint                      | regclass                    | oid                | yes
         |
bigint                      | regoper                     | oid                | yes
         |
bigint                      | regoperator                 | oid                | yes
         |
bigint                      | regproc                     | oid                | yes
         |
bigint                      | regprocedure                | oid                | yes
         |
bigint                      | regtype                     | oid                | yes
         |
bigint                      | smallint                    | int2               | in
assignment |
bigint                      | super                       | int8_partiql       | in
assignment |
```

\dd[S] [PATTERN]

Shows object descriptions not displayed elsewhere.

\de

Lists external tables. This includes tables in the AWS Glue Data Catalog, Hive Metastore and federated tables from Amazon RDS/Aurora MySQL, Amazon RDS/Aurora PostgreSQL and Amazon Redshift datashare tables.

\de NAME

Describes an external table.

The following example shows an AWS Glue external table.

```
# \de spectrum.lineitem
```

```
                          Glue External table "spectrum.lineitem"
     Column      | External Type | Redshift Type | Position | Partition Key | Nullable
-----------------+---------------+---------------+----------+---------------+----------
 l_orderkey      | bigint        | bigint        | 1        | 0             |
 l_partkey       | bigint        | bigint        | 2        | 0             |
 l_suppkey       | int           | int           | 3        | 0             |
 l_linenumber    | int           | int           | 4        | 0             |
 l_quantity      | decimal(12,2) | decimal(12,2) | 5        | 0             |
 l_extendedprice | decimal(12,2) | decimal(12,2) | 6        | 0             |
 l_discount      | decimal(12,2) | decimal(12,2) | 7        | 0             |
 l_tax           | decimal(12,2) | decimal(12,2) | 8        | 0             |
 l_returnflag    | char(1)       | char(1)       | 9        | 0             |
 l_linestatus    | char(1)       | char(1)       | 10       | 0             |
 l_shipdate      | date          | date          | 11       | 0             |
 l_commitdate    | date          | date          | 12       | 0             |
 l_receiptdate   | date          | date          | 13       | 0             |
 l_shipinstruct  | char(25)      | char(25)      | 14       | 0             |
 l_shipmode      | char(10)      | char(10)      | 15       | 0             |
 l_comment       | varchar(44)   | varchar(44)   | 16       | 0             |

Location: s3://redshiftbucket/kfhose2019/12/31
Input_format: org.apache.hadoop.mapred.TextInputFormat
Output_format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Serialization_lib: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
Serde_parameters: {"field.delim":"|","serialization.format":"|"}
Parameters:
 {"EXTERNAL":"TRUE","numRows":"178196721475","transient_lastDdlTime":"1577771873"}
```

A Hive Metastore table.

```
# \de emr.lineitem
                    Hive Metastore External Table "emr.lineitem"
     Column      | External Type | Redshift Type | Position | Partition Key | Nullable
-----------------+---------------+---------------+----------+---------------+----------
 l_orderkey      | bigint        | bigint        | 1        | 0             |
 l_partkey       | bigint        | bigint        | 2        | 0             |
 l_suppkey       | int           | int           | 3        | 0             |
 l_linenumber    | int           | int           | 4        | 0             |
 l_quantity      | decimal(12,2) | decimal(12,2) | 5        | 0             |
 l_extendedprice | decimal(12,2) | decimal(12,2) | 6        | 0             |
 l_discount      | decimal(12,2) | decimal(12,2) | 7        | 0             |
 l_tax           | decimal(12,2) | decimal(12,2) | 8        | 0             |
```

```
 l_returnflag    | char(1)       | char(1)       | 9        | 0            |
 l_linestatus    | char(1)       | char(1)       | 10       | 0            |
 l_commitdate    | date          | date          | 11       | 0            |
 l_receiptdate   | date          | date          | 12       | 0            |
 l_shipinstruct  | char(25)      | char(25)      | 13       | 0            |
 l_shipmode      | char(10)      | char(10)      | 14       | 0            |
 l_comment       | varchar(44)   | varchar(44)   | 15       | 0            |
 l_shipdate      | date          | date          | 16       | 1            |

Location: s3://redshiftbucket/cetas
Input_format: org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat
Output_format: org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat
Serialization_lib: org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe
Serde_parameters: {"serialization.format":"1"}
Parameters: {"EXTERNAL":"TRUE", "numRows":"4307207",
 "transient_lastDdlTime":"1626990007"}
```

PostgreSQL external table.

```
# \de pgrsql.alltypes
                              Postgres Federated Table "pgrsql.alltypes"
 Column |        External Type        |        Redshift Type        | Position |
 Partition Key | Nullable
--------+-----------------------------+-----------------------------+----------
+---------------+----------
 col1   | bigint                      | bigint                      | 1        | 0
        |
 col2   | bigint                      | bigint                      | 2        | 0
        |
 col5   | boolean                     | boolean                     | 3        | 0
        |
 col6   | box                         | varchar(65535)              | 4        | 0
        |
 col7   | bytea                       | varchar(65535)              | 5        | 0
        |
 col8   | character(10)               | character(10)               | 6        | 0
        |
 col9   | character varying(10)       | character varying(10)       | 7        | 0
        |
 col10  | cidr                        | varchar(65535)              | 8        | 0
        |
```

```
 col11  | circle                      | varchar(65535)              | 9        | 0
        |
 col12  | date                        | date                        | 10       | 0
        |
 col13  | double precision            | double precision            | 11       | 0
        |
 col14  | inet                        | varchar(65535)              | 12       | 0
        |
 col15  | integer                     | integer                     | 13       | 0
        |
 col16  | interval                    | varchar(65535)              | 14       | 0
        |
 col17  | json                        | varchar(65535)              | 15       | 0
        |
 col18  | jsonb                       | varchar(65535)              | 16       | 0
        |
 col19  | line                        | varchar(65535)              | 17       | 0
        |
 col20  | lseg                        | varchar(65535)              | 18       | 0
        |
 col21  | macaddr                     | varchar(65535)              | 19       | 0
        |
 col22  | macaddr8                    | varchar(65535)              | 20       | 0
        |
 col23  | money                       | varchar(65535)              | 21       | 0
        |
 col24  | numeric                     | numeric(38,20)              | 22       | 0
        |
 col25  | path                        | varchar(65535)              | 23       | 0
        |
 col26  | pg_lsn                      | varchar(65535)              | 24       | 0
        |
 col28  | point                       | varchar(65535)              | 25       | 0
        |
 col29  | polygon                     | varchar(65535)              | 26       | 0
        |
 col30  | real                        | real                        | 27       | 0
        |
 col31  | smallint                    | smallint                    | 28       | 0
        |
 col32  | smallint                    | smallint                    | 29       | 0
        |
 col33  | integer                     | integer                     | 30       | 0
        |
```

```
 col34  | text                        | varchar(65535)              | 31       | 0
        |
 col35  | time without time zone      | varchar(65535)              | 32       | 0
        |
 col36  | time with time zone         | varchar(65535)              | 33       | 0
        |
 col37  | timestamp without time zone | timestamp without time zone | 34       | 0
        |
 col38  | timestamp with time zone    | timestamp with time zone    | 35       | 0
        |
 col39  | tsquery                     | varchar(65535)              | 36       | 0
        |
 col40  | tsvector                    | varchar(65535)              | 37       | 0
        |
 col41  | txid_snapshot               | varchar(65535)              | 38       | 0
        |
 col42  | uuid                        | varchar(65535)              | 39       | 0
        |
 col43  | xml                         | varchar(65535)              | 40       | 0
        |
```

\df[anptw][S+] [PATTERN]

Lists functions of various types. The command `\df`, for example, returns a list of functions. Results include properties like name, data-type returned, access privileges, and additional metadata. Function types can include triggers, stored procedures, window functions and other types. When you append S+ to the command, for example `\dfantS+`, additional metadata columns are included, such as `owner`, `security`, and `access privileges`.

\dL[S+] [PATTERN]

Lists data about procedural languages associated with the database. Information includes the name, such as plpgsql, and additional metadata, which includes whether it is trusted, access privileges, and description. Sample call is, for example, `\dLS+`, which lists languages and their properties. When you append S+ to the command, additional metadata columns are included, such as `call handler` and `access privileges`.

Sample results:

```
List of languages
```

```
   name    | trusted | internal language |      call handler       |
      validator                         | access privileges |          description

-----------+---------+------------------+-------------------------
+-----------------------------------------------------------+-------------------
+--------------------------------
 c         | f       | t                | -                       |
 fmgr_c_validator(oid)                                       |                 |
 Dynamically-loaded C functions
 exfunc    | f       | f                | exfunc_call_handler()   | -
                                        | rdsdb=U/rdsdb     |
 internal  | f       | t                | -                       |
 fmgr_internal_validator(oid)                                |                 |
 Built-in functions
 mlfunc    | f       | f                | mlfunc_call_handler()   | -
                                        | rdsdb=U/rdsdb     |
 plpgsql   | t       | f                | plpgsql_call_handler()  |
 plpgsql_validator(oid)                                      |                 |
 plpythonu | f       | f                | plpython_call_handler() |
 plpython_compiler(cstring,cstring,cstring,cstring,cstring) | rdsdb=U/rdsdb     |
 sql       | t       | t                | -                       |
 fmgr_sql_validator(oid)                                     | =U/rdsdb          | SQL-
language functions
```

\dm[S+] [PATTERN]

Lists materialized views. For example, `\dmS+` lists materialized views and their properties. When you append S+ to the command, additional metadata columns are included.

\dn[S+] [PATTERN]

Lists schemas. When you append S+ to the command, for example `\dnS+`, additional metadata columns are included, such as `description` and `access privileges`.

\dp [PATTERN]

Lists table, view, and sequence access privileges.

\dt[S+] [PATTERN]

Lists tables. When you append S+ to the command, for example `\dtS+`, additional metadata columns are included, such as `description` in this case.

\du

Lists the users for the database. Includes their name and their roles, such as Superuser, and attributes.

\dv[S+] [PATTERN]

Lists views. Includes schema, type, and owner data. When you append S+ to the command, for example `\dvS+`, additional metadata columns are included.

\H

Turns on HTML output. This is useful to quickly return formatted results. For example, `select * from sales; \H` returns results from the sales table, in HTML. To switch back to tablular results, use `\q`, or quiet.

\i

Runs commands from a file. For example, assuming you have rsql_steps.sql in your working directory, the following runs the commands in the file: `\i rsql_steps.sql`.

\l[+] [PATTERN]

Lists databases. Includes owner, encoding, and additional information.

\q

The quit, or `\q` command, logs off database sessions and exits RSQL.

\sv[+] VIEWNAME

Shows a view's definition.

\timing

Shows the run time, for a query, for instance.

\z [PATTERN]

The same output as \dp.

\?

Shows help information. The optional parameter specifies the item to explain.

\EXIT

Logs off all database sessions and exits Amazon Redshift RSQL. In addition, you can specify an optional exit code. For example, `\EXIT 15` will exit the Amazon Redshift RSQL terminal and return exit code 15.

The following example shows output from a connection and exit from RSQL.

```
% rsql -D testuser
DSN Connected
DBMS Name: Amazon Redshift
Driver Name: Amazon Redshift ODBC Driver
Driver Version: 1.4.34.1000
Rsql Version: 1.0.1
Redshift Version: 1.0.29306
Type "help" for help.

(testcluster) user1@dev=# \exit 15

% echo $?
15
```

\EXPORT

Specifies the name of an export file that RSQL uses to store database information returned by a subsequent SQL SELECT statement.

export_01.sql

```
\export report file='E:\\accounts.out'
\rset rformat off
\rset width 1500
\rset heading "General Title"
\rset titledashes on
```

```
select * from td_dwh.accounts;
\export reset
```

Console output

```
Rformat is off.
Target width is 1500.
Heading is set to: General Title
Titledashes is on.
(exported 40 rows)
```

\LOGON

Connects to a database. You can specify connection parameters using positional syntax or as a connection string.

Command syntax is the following: `\logon {[DBNAME|- USERNAME|- HOST|- PORT|- [PASSWORD]] | conninfo}`

The `DBNAME` is the name of the database to connect to. The `USERNAME` is the user name to connect as. The default `HOST` is `localhost`. The default `PORT` is `5439`.

When a host name is specified in a `\LOGON` command, it becomes the default host name for additional `\LOGON` commands. To change the default host name, specify a new `HOST` in an additional `\LOGON` command.

Sample output from the `\LOGON` command for `user1` follows.

```
(testcluster) user1@redshiftdb=# \logon dev
DBMS Name: Amazon Redshift
Driver Name: Amazon Redshift ODBC Driver
Driver Version: 1.4.27.1000
Rsql Version: 1.0.1
You are now connected to database "dev" as user "user1".
(testcluster) user1@dev=#
```

Sample output for *user2*.

```
(testcluster) user1@dev=# \logon dev user2 testcluster2.example.com
Password for user user2:
```

```
DBMS Name: Amazon Redshift
Driver Name: Amazon Redshift ODBC Driver
Driver Version: 1.4.27.1000
Rsql Version: 1.0.1
You are now connected to database "dev" as user "user2" on host
 "testcluster2.example.com" at port "5439".
(testcluster2) user2@dev=#
```

\REMARK

An extension of the \echo command. \REMARK prints the specified string to the output stream. \REMARK extends \echo by adding the ability to break the output over separate lines.

The following sample shows output from the command.

```
(testcluster) user1@dev=# \remark 'hello//world'
hello
world
```

\RSET

The command \rset sets command parameters and variables. \rset has both an interactive and a batch mode. It doesn't support options as bash options, like *-x*, or arguments, for instance *--<arg>*.

It sets variables, such as the following:

- ERRORLEVEL
- HEADING and RTITLE
- RFORMAT
- MAXERROR
- TITLEDASHES
- WIDTH

The following example specifies a heading.

```
\rset heading "Winter Sales Report"
```

For more examples of how to use \rset, you can find several in the Amazon Redshift RSQL variables topics.

\RUN

Runs the Amazon Redshift RSQL script contained in the specified file. \RUN extends the \i command by adding an option to skip header lines in a file.

If the file name includes a comma, semicolon, or space, enclose it in single quotation marks. Additionally, if text follows the file name, enclose it in quotation marks. In UNIX, file names are case sensitive. In Windows, file names are case insensitive.

The following sample shows output from the command.

```
(testcluster) user1@dev=# \! cat test.sql
select count(*) as lineitem_cnt from lineitem;
select count(*) as customer_cnt from customer;
select count(*) as orders_cnt from orders;



(testcluster) user1@dev=# \run file=test.sql
 lineitem_cnt
--------------
      4307207
(1 row)

 customer_cnt
--------------
     37796166
(1 row)

 orders_cnt
------------
          0
(1 row)


(testcluster) user1@dev=# \run file=test.sql skip=2
2 records skipped in RUN file.
 orders_cnt
------------
          0
```

```
(1 row)
```

\OS

An alias for the \! command. \OS runs the operating system command that is passed as a parameter. Control returns to Amazon Redshift RSQL after the command is run. For example, you can run the following command to print the current system date time and return to the RSQL terminal: \os date.

```
(testcluster) user1@dev=# \os date
Tue Sep 7 20:47:54 UTC 2021
```

\GOTO

A new command for Amazon Redshift RSQL. \GOTO skips all intervening commands and resumes processing at the specified \LABEL. The \LABEL must be a forward reference. You cannot jump to a \LABEL that lexically precedes the \GOTO.

The following shows sample output.

```
(testcluster) user1@dev=# \! cat test.sql
select count(*) as cnt from lineitem \gset
select :cnt as cnt;
\if :cnt > 100
    \goto LABELB
\endif

\label LABELA
\remark 'this is label LABELA'
\label LABELB
\remark 'this is label LABELB'


(testcluster) user1@dev=# \i test.sql
   cnt
---------
 4307207
(1 row)

\label LABELA ignored
\label LABELB processed
```

```
this is label LABELB
```

\LABEL

A new command for Amazon Redshift RSQL. \LABEL establishes an entry point for running the program, as the target for a \GOTO command.

The following shows sample output from the command.

```
(testcluster) user1@dev=# \! cat test.sql
select count(*) from lineitem limit 5;
\goto LABELB
\remark "this step was skipped by goto label";
\label LABELA
\remark 'this is label LABELA'
\label LABELB
\remark 'this is label LABELB'



(testcluster) user1@dev=# \i testgoto.sql
  count
 4307193
(1 row)

\label LABELA ignored
\label LABELB processed
this is label LABELB
```

\IF (\ELSEIF, \ELSE, \ENDIF)

\IF and related commands conditionally run portions of the input script. An extension of the PSQL \if (\elif, \else, \endif) command. \IF and \ELSEIF support boolean expressions including AND, OR and NOT conditions.

The following shows sample output from the commands.

```
(testcluster) user1@dev=# \! cat test.sql
SELECT query FROM stv_inflight LIMIT 1 \gset
select :query as query;
\if :query > 1000000
    \remark 'Query id is greater than 1000000'
```

```
\elseif :query = 1000000
    \remark 'Query id is equal than 1000000'
\else
    \remark 'Query id is less than 1000000'
\endif


(testcluster) user1@dev=# \i test.sql
 query
--------
 994803
(1 row)

Query id is less than 1000000
```

Use ERRORCODE in your branching logic.

```
\if :'ERRORCODE' = '00000'
    \remark 'The statement was executed without error'
\else
    \remark :LAST_ERROR_MESSAGE
\endif
```

Use \GOTO within an \IF block to control how code is run.

Amazon Redshift RSQL variables

Some keywords act as variables in RSQL. You can set each to a specific value, or re-set the value. Most are set with \rset, which has an interactive mode and a batch mode. Commands may be defined in lower or upper case.

ACTIVITYCOUNT

Indicates the number of rows affected by the last submitted request. For a data-returning request, this is the number of rows returned to RSQL from the database. The value is 0 or a positive integer. The maximum value is 18,446,744,073,709,551,615.

The specially treated variable ACTIVITYCOUNT is similar to the variable ROW_COUNT. However, ROW_COUNT doesn't report a count of affected rows to the client application at command completion for SELECT, COPY or UNLOAD. But ACTIVITYCOUNT does.

activitycount_01.sql:

```
select viewname, schemaname
from pg_views
where schemaname = 'not_existing_schema';
\if :ACTIVITYCOUNT = 0
\remark 'views do not exist'
\endif
```

Console output:

```
viewname | schemaname
----------+------------
(0 rows)

views do not exist
```

ERRORLEVEL

Assigns severity levels to errors. Use the severity levels to determine a course of action. If the ERRORLEVEL command has not been used, its value is ON by default.

errorlevel_01.sql:

```
\rset errorlevel 42P01 severity 0

select * from tbl;

select 1 as col;

\echo exit
\quit
```

Console output:

```
Errorlevel is on.
rsql: ERROR: relation "tbl" does not exist
(1 row)

col
1

exit
```

HEADING and RTITLE

Enables users to specify a header that appears at the top of a report. Header specified by the RSET RTITLE command automatically includes the current system date of the client computer.

rset_heading_rtitle_02.rsql content:

```
\remark Starting...
\rset rtitle "Marketing Department||Confidential//Third Quarter//Chicago"
\rset width 70
\rset rformat on
select * from rsql_test.tbl_currency order by id limit 2;
\exit
\remark Finishing...
```

Console output:

```
Starting...
Rtitle is set to: &DATE||Marketing Department||Confidential//Third Quarter//Chicago
 (Changes will take effect after RFORMAT is
switched ON)
Target width is 70.
Rformat is on.
09/11/20       Marketing       Department Confidential
                  Third Quarter
                     Chicago
id  | bankid  | name |      start_date
100 |       1 | USD | 2020-09-11 10:51:39.106905
110 |       1 | EUR | 2020-09-11 10:51:39.106905
(2 rows)

Press any key to continue . . .
```

MAXERROR

Designates a maximum error-severity level beyond which RSQL terminates job processing. Return codes are integer values that RSQL returns to the client operating system after completing each job or task. The value of the return code indicates the completion status of the job or task. If a script contains a statement that produces an error-severity level greater than the designated `maxerror` value, RSQL immediately exits. Therefore, to have RSQL exit on an error-severity level of 8, use `RSET MAXERROR 7`.

maxerror_01.sql content:

```
\rset maxerror 0

select 1 as col;

\quit
```

Console output:

```
Maxerror is default.
(1 row)

col
1
```

RFORMAT

Enables users to specify whether to apply settings for the formatting commands.

rset_rformat.rsql content:

```
\remark Starting...
\pset border 2
\pset format wrapped
\pset expanded on
\pset title 'Great Title'
select * from rsql_test.tbl_long where id = 500;
\rset rformat
select * from rsql_test.tbl_long where id = 500;
\rset rformat off
select * from rsql_test.tbl_long where id = 500;
\rset rformat on
select * from rsql_test.tbl_long where id = 500;
\exit
\remark Finishing...
```

Console output:

```
Starting...
Border style is 2. (Changes will take effect after RFORMAT is switched ON)
Output format is wrapped. (Changes will take effect after RFORMAT is switched ON)
```

```
Expanded display is on. (Changes will take effect after RFORMAT is switched ON)
Title is "Great Title". (Changes will take effect after RFORMAT is switched ON)
id  |                                                                  long_string
500 | In general, the higher the number the more borders and lines the tables will
 have, but details depend on the particular
format.
(1 row)

Rformat is on.
Great Title
+-[ RECORD
 1 ]+-------------------------------------------------------------------------------------------
-----------+
| id          | 500
|
| long_string | In general, the higher the number the more borders and lines the tables
 will have, but details depend on the
particular format. |
+-------------
+-------------------------------------------------------------------------------------------
-----------+

Rformat is off.
id  |                                                                  long_string
500 | In general, the higher the number the more borders and lines the tables will
 have, but details depend on the particular format.
(1 row)

Rformat is on.
Great Title
+-[ RECORD
 1 ]+-------------------------------------------------------------------------------------------
-----------+
| id          | 500
|
| long_string | In general, the higher the number the more borders and lines the tables
 will have, but details depend on the
particular format. |
+-------------
+-------------------------------------------------------------------------------------------
-----------+
Press any key to continue . . .
```

ROW_COUNT

Gets the number of records affected by the previous query. It's typically used to check a result, like in the following code fragment:

```
SET result = ROW_COUNT;

IF result = 0
...
```

TITLEDASHES

This control enables users to specify whether a line of dash characters is to be printed above the column data returned for SQL statements.

Example:

```
\rset titledashes on
select dept_no, emp_no, salary from rsql_test.EMPLOYEE
where dept_no = 100;
\rset titledashes off
select dept_no, emp_no, salary from rsql_test.EMPLOYEE
where dept_no = 100;
```

Console output:

```
dept_no      emp_no          salary
------------ ------------ --------------------
100          1000346         1300.00
100          1000245         5000.00
100          1000262         2450.00

dept_no      emp_no          salary
100          1000346         1300.00
100          1000245         5000.00
100          1000262         2450.00
```

WIDTH

Sets the output format to wrapped and specifies the target width for each line in a report. Without a parameter, it returns the current settings for both the format and target width.

rset_width_01.rsql content:

```
\echo Starting...
\rset width
\rset width 50
\rset width
\quit
\echo Finishing...
```

Console output:

```
Starting...
Target width is 75.
Target width is 50.
Target width is 50.
Press any key to continue . . .
```

Example with parameter:

```
\echo Starting...
\rset rformat on
\pset format wrapped
select * from rsql_test.tbl_long where id = 500;
\rset width 50
select * from rsql_test.tbl_long where id = 500;
\quit
\echo Finishing...
```

Console output:

```
Starting...
Rformat is on.
Output format is wrapped.
id  |                                          long_string
500 | In general, the higher the number the more borders and lines the ta.
    |.bles will have, but details depend on the particular format.
(1 row)

Target width is 50.
id  |                                          long_string
500 | In general, the higher the number the more.
```

```
    |. borders and lines the tables will have, b.
    |.ut details depend on the particular format.
    |..
(1 row)
Press any key to continue . . .
```

Amazon Redshift RSQL error codes

Success messages, warnings, and exceptions:

Error Code	Error Class	Condition Name
00000	Class 00 — Successful Completion	successful_completion
01000	Class 01 — Warning	warning
0100C	Class 01 — Warning	dynamic_result_sets_returned
01008	Class 01 — Warning	implicit_zero_bit_padding
01003	Class 01 — Warning	null_value_eliminated_in_set_function
01007	Class 01 — Warning	privilege_not_granted
01006	Class 01 — Warning	privilege_not_revoked
01004	Class 01 — Warning	string_data_right_truncation
01P01	Class 01 — Warning	deprecated_feature
02000	Class 02 — No Data	no_data
02001	Class 02 — No Data	no_additional_dynamic_result_sets_returned
03000	Class 03 — SQL Statement Not Yet Complete	sql_statement_not_yet_complete
08000	Class 08 — Connection Exception	connection_exception

Error Code	Error Class	Condition Name
08003	Class 08 — Connection Exception	connection_does_not_exist
08006	Class 08 — Connection Exception	connection_failure
08001	Class 08 — Connection Exception	sqlclient_unable_to_establish_sqlconnection
08004	Class 08 — Connection Exception	sqlserver_rejected_establishment_of_sqlconnection
08007	Class 08 — Connection Exception	transaction_resolution_unknown
08P01	Class 08 — Connection Exception	protocol_violation
09000	Class 09 — Triggered Action Exception	triggered_action_exception
0A000	Class 0A — Feature Not Supported	feature_not_supported
0A000	Class 0A — Feature Not Supported	feature_not_supported
0B000	Class 0B — Invalid Transaction Initiation	invalid_transaction_initiation
0F000	Class 0F — Locator Exception	locator_exception
0F001	Class 0F — Locator Exception	invalid_locator_specification
0L000	Class 0L — Invalid Grantor	invalid_grantor
0LP01	Class 0L — Invalid Grantor	invalid_grant_operation

Error Code	Error Class	Condition Name
0P000	Class 0P — Invalid Role Specification	invalid_role_specification
0Z000	Class 0Z — Diagnostics Exception	diagnostics_exception
0Z002	Class 0Z — Diagnostics Exception	stacked_diagnostics_accessed_without_active_handler
20000	Class 20 — Case Not Found	case_not_found
21000	Class 21 — Cardinality Violation	cardinality_violation

Data exceptions:

Error Code	Error Class	Condition Name
22000	Class 22 — Data Exception	data_exception
2202E	Class 22 — Data Exception	array_subscript_error
22021	Class 22 — Data Exception	character_not_in_repertoire
22008	Class 22 — Data Exception	datetime_field_overflow
22012	Class 22 — Data Exception	division_by_zero
22005	Class 01 — Warning	error_in_assignment
2200B	Class 01 — Warning	escape_character_conflict
22022	Class 01 — Warning	indicator_overflow
22015	Class 01 — Warning	interval_field_overflow
2201E	Class 01 — Warning	invalid_argument_for_logarithm

Error Code	Error Class	Condition Name
2201F	Class 01 — Warning	invalid_argument_for_power_function
2201G	Class 01 — Warning	invalid_argument_for_width_bucket_function
22018	Class 01 — Warning	invalid_character_value_for_cast
22007	Class 01 — Warning	invalid_datetime_format
22019	Class 01 — Warning	invalid_escape_character
2200D	Class 01 — Warning	invalid_escape_octet
22025	Class 01 — Warning	invalid_escape_sequence
22P06	Class 01 — Warning	nonstandard_use_of_escape_character
22010	Class 01 — Warning	invalid_indicator_parameter_value
22023	Class 01 — Warning	invalid_parameter_value
2201B	Class 01 — Warning	invalid_regular_expression
22009	Class 01 — Warning	invalid_time_zone_displacement_value
2200C	Class 01 — Warning	invalid_use_of_escape_character
2200G	Class 01 — Warning	most_specific_type_mismatch
22004	Class 01 — Warning	null_value_not_allowed
22002	Class 01 — Warning	null_value_no_indicator_parameter
22003	Class 01 — Warning	numeric_value_out_of_range
22026	Class 01 — Warning	string_data_length_mismatch
22001	Class 01 — Warning	string_data_right_truncation
22011	Class 01 — Warning	substring_error

Error Code	Error Class	Condition Name
22027	Class 01 — Warning	trim_error
22024	Class 01 — Warning	unterminated_c_string
2200F	Class 01 — Warning	zero_length_character_string
22P01	Class 01 — Warning	floating_point_exception
22P02	Class 01 — Warning	invalid_text_representation
22P03	Class 01 — Warning	invalid_binary_representation
22P04	Class 01 — Warning	bad_copy_file_format
22P05	Class 01 — Warning	untranslatable_character

Integrity constraint violations:

Error Code	Error Class	Condition Name
23000	Class 23 — Integrity Constraint Violation	integrity_constraint_violation
23001	Class 23 — Integrity Constraint Violation	restrict_violation
23502	Class 23 — Integrity Constraint Violation	not_null_violation
23503	Class 23 — Integrity Constraint Violation	foreign_key_violation
23505	Class 23 — Integrity Constraint Violation	unique_violation
23514	Class 23 — Integrity Constraint Violation	check_violation

Error Code	Error Class	Condition Name
24000	Class 24 — Invalid Cursor State	invalid_cursor_state
01004	Class 01 — Warning	string_data_right_truncation
25000	Class 25 — Invalid Transaction State	invalid_transaction_state
25001	Class 25 — Invalid Transaction State	active_sql_transaction
25002	Class 25 — Invalid Transaction State	invalid_transaction_state
25008	Class 25 — Invalid Transaction State	held_cursor_requires_same_isolation_level
25003	Class 25 — Invalid Transaction State	inappropriate_access_mode_for_branch_transaction
25004	Class 25 — Invalid Transaction State	inappropriate_isolation_level_for_branch_transaction
25005	Class 25 — Invalid Transaction State	no_active_sql_transaction_for_branch_transaction
25006	Class 25 — Invalid Transaction State	read_only_sql_transaction
25007	Class 25 — Invalid Transaction State	no_active_sql_transaction_for_branch_transaction
25P01	Class 25 — Invalid Transaction State	no_active_sql_transaction
25P02	Class 25 — Invalid Transaction State	in_failed_sql_transaction

Error Code	Error Class	Condition Name
26000	Class 26 — Invalid SQL Statement Name	invalid_sql_statement_name
28000	Class 28 — Invalid Authorization Specification	invalid_authorization_specification
2B000	Class 2B — Dependent Privilege Descriptors Still Exist	dependent_privilege_descriptors_still_exist
2BP01	Class 2B — Dependent Privilege Descriptors Still Exist	dependent_objects_still_exist
2D000	Class 2D — Invalid Transaction Termination	invalid_transaction_termination
2F000	Class 2F — SQL Routine Exception	sql_routine_exception
2F005	Class 2F — SQL Routine Exception	function_executed_no_return_statement
2F002	Class 2F — SQL Routine Exception	modifying_sql_data_not_permitted
2F003	Class 2F — SQL Routine Exception	prohibited_sql_statement_attempted
2F004	Class 2F — SQL Routine Exception	reading_sql_data_not_permitted
34000	Class 34 — Invalid Cursor Name	invalid_cursor_name
38000	Class 38 — External Routine Exception	external_routine_exception
38001	Class 38 — External Routine Exception	containing_sql_not_permitted

Error Code	Error Class	Condition Name
38002	Class 38 — External Routine Exception	modifying_sql_data_not_permitted
38003	Class 38 — External Routine Exception	prohibited_sql_statement_attempted
38004	Class 38 — External Routine Exception	reading_sql_data_not_permitted
39000	Class 39 — External Routine Invocation Exception	external_routine_invocation_exception
39001	Class 39 — External Routine Invocation Exception	invalid_sqlstate_returned
39004	Class 39 — External Routine Invocation Exception	null_value_not_allowed
39P01	Class 39 — External Routine Invocation Exception	trigger_protocol_violated
39P02	Class 39 — External Routine Invocation Exception	srf_protocol_violated
3D000	Class 3D — Invalid Catalog Name	invalid_catalog_name
3F000	Class 3F — Invalid Schema Name	invalid_schema_name
42000	Class 42 — Syntax Error or Access Rule Violation	syntax_error_or_access_rule_violation
42601	Class 42 — Syntax Error or Access Rule Violation	syntax_error
42501	Class 42 — Syntax Error or Access Rule Violation	insufficient_privilege

Error Code	Error Class	Condition Name
42846	Class 42 — Syntax Error or Access Rule Violation	cannot_coerce
42803	Class 42 — Syntax Error or Access Rule Violation	grouping_error
42830	Class 42 — Syntax Error or Access Rule Violation	invalid_foreign_key
42602	Class 42 — Syntax Error or Access Rule Violation	invalid_name
42622	Class 42 — Syntax Error or Access Rule Violation	name_too_long
42939	Class 42 — Syntax Error or Access Rule Violation	reserved_name
42804	Class 42 — Syntax Error or Access Rule Violation	datatype_mismatch
42P18	Class 42 — Syntax Error or Access Rule Violation	indeterminate_datatype
42809	Class 42 — Syntax Error or Access Rule Violation	wrong_object_type
42703	Class 42 — Syntax Error or Access Rule Violation	undefined_column
42883	Class 42 — Syntax Error or Access Rule Violation	undefined_function
42P01	Class 42 — Syntax Error or Access Rule Violation	undefined_table
42P02	Class 42 — Syntax Error or Access Rule Violation	undefined_parameter

Error Code	Error Class	Condition Name
42704	Class 42 — Syntax Error or Access Rule Violation	undefined_object
42701	Class 42 — Syntax Error or Access Rule Violation	duplicate_column
42P03	Class 42 — Syntax Error or Access Rule Violation	duplicate_cursor
42P04	Class 42 — Syntax Error or Access Rule Violation	duplicate_database
42723	Class 42 — Syntax Error or Access Rule Violation	duplicate_function
42P05	Class 42 — Syntax Error or Access Rule Violation	duplicate_prepared_statement
42P06	Class 42 — Syntax Error or Access Rule Violation	duplicate_schema
42P07	Class 42 — Syntax Error or Access Rule Violation	duplicate_table
42712	Class 42 — Syntax Error or Access Rule Violation	duplicate_alias
42710	Class 42 — Syntax Error or Access Rule Violation	duplicate_object
42702	Class 42 — Syntax Error or Access Rule Violation	ambiguous_column
42725	Class 42 — Syntax Error or Access Rule Violation	ambiguous_function
42P08	Class 42 — Syntax Error or Access Rule Violation	ambiguous_parameter

Error Code	Error Class	Condition Name
42P09	Class 42 — Syntax Error or Access Rule Violation	ambiguous_alias
42P10	Class 42 — Syntax Error or Access Rule Violation	invalid_column_reference
42611	Class 42 — Syntax Error or Access Rule Violation	invalid_column_definition
42P11	Class 42 — Syntax Error or Access Rule Violation	invalid_cursor_definition
42P12	Class 42 — Syntax Error or Access Rule Violation	invalid_database_definition
42P13	Class 42 — Syntax Error or Access Rule Violation	invalid_function_definition
42P14	Class 42 — Syntax Error or Access Rule Violation	invalid_prepared_statement_definition
42P15	Class 42 — Syntax Error or Access Rule Violation	invalid_schema_definition
42P16	Class 42 — Syntax Error or Access Rule Violation	invalid_table_definition
42P17	Class 42 — Syntax Error or Access Rule Violation	invalid_object_definition
44000	Class 44 — WITH CHECK OPTION Violation	with_check_option_violation
53000	Class 53 — Insufficient Resources	insufficient_resources
53100	Class 53 — Insufficient Resources	disk_full

Error Code	Error Class	Condition Name
53200	Class 53 — Insufficient Resources	out_of_memory
53300	Class 53 — Insufficient Resources	too_many_connections
54000	Class 54 — Program Limit Exceeded	program_limit_exceeded
54001	Class 54 — Program Limit Exceeded	statement_too_complex
54011	Class 54 — Program Limit Exceeded	too_many_columns
54023	Class 54 — Program Limit Exceeded	too_many_arguments
55000	Class 55 — Object Not In Prerequisite State	object_not_in_prerequisite_state
55006	Class 55 — Object Not In Prerequisite State	object_in_use
55P02	Class 55 — Object Not In Prerequisite State	cant_change_runtime_param
55P03	Class 55 — Object Not In Prerequisite State	lock_not_available
57000	Class 57 — Operator Intervention	operator_intervention
57014	Class 57 — Operator Intervention	query_canceled
57P01	Class 57 — Operator Intervention	admin_shutdown

Error Code	Error Class	Condition Name
57P02	Class 57 — Operator Intervention	crash_shutdown
57P03	Class 57 — Operator Intervention	cannot_connect_now
58000	Class 58 — System Error (errors external to PostgreSQL)	system_error
58030	Class 58 — System Error (errors external to PostgreSQL)	io_error
58P01	Class 58 — System Error (errors external to PostgreSQL)	undefined_file
58P02	Class 58 — System Error (errors external to PostgreSQL)	duplicate_file
F0000	Class F0 — Configuration File Error	duplicate_file
F0001	Class F0 — Configuration File Error	lock_file_exists
P0000	Class P0 — PL/pgSQL Error	plpgsql_error
P0001	Class P0 — PL/pgSQL Error	raise_exception
P0002	Class P0 — PL/pgSQL Error	no_data_found
P0003	Class P0 — PL/pgSQL Error	too_many_rows
XX000	Class XX — Internal Error	internal_error

Error Code	Error Class	Condition Name
XX001	Class XX — Internal Error	data_corrupted
XX002	Class XX — Internal Error	index_corrupted

Amazon Redshift RSQL environment variables

Amazon Redshift RSQL can use environment variables to select default parameter values.

RSPASSWORD

> **Important**
>
> We don't recommend using this environment variable for security reasons, as some operating systems allow non-administrative users to see process environment variables.

Sets the password for Amazon Redshift RSQL to use when connecting to Amazon Redshift. This environment variable requires Amazon Redshift RSQL 1.0.4 and above.

RSQL prioritizes RSPASSWORD if one is set. If RSPASSWORD is not set and you're connecting using a DSN, RSQL takes the password from the DSN file's parameters. Finally, if RSPASSWORD is not set and you're not using a DSN, RSQL provides a password prompt after attempting to connect.

The following is an example of setting an RSPASSWORD:

```
export RSPASSWORD=TestPassw0rd
```

Using an authentication profile to connect to Amazon Redshift

If you have many connections to Amazon Redshift, it can be difficult to manage settings for all of them. Often, each JDBC or ODBC connection uses specific configuration options. By using an authentication profile, you can store connection options together. This way, your users can choose a profile to connect with and avoid managing settings for individual options. Profiles can apply to various scenarios and user types.

After you create an authentication profile, users can add the ready-to-use profile to a connection string. By doing this, they can connect to Amazon Redshift with the right settings for each role and use case.

For Amazon Redshift API information, see CreateAuthenticationProfile.

Creating an authentication profile

Using the AWS CLI, you create an authentication profile with the `create-authentication-profile` command. This assumes that you have an existing Amazon Redshift cluster and an existing database. Your credentials must have permission to connect to the Amazon Redshift database and rights to fetch the authentication profile. You provide the configuration options as a JSON string, or reference a file containing your JSON string.

```
create-authentication-profile --authentication-profile-name<value: String> --
authentication-profile-content<value: String>
```

The following example creates a profile called `ExampleProfileName`. Here, you can add keys and values that define your cluster name and other option settings, as a JSON string.

```
create-authentication-profile --authentication-profile-name "ExampleProfileName"
 --authentication-profile-content "{\"AllowDBUserOverride\":\"1\",\"Client_ID
\":\"ExampleClientID\",\"App_ID\":\"ExampleAppID\",\"AutoCreate\":false,
\"enableFetchRingBuffer\":true,\"databaseMetadataCurrentDbOnly\":true}"
}
```

This command creates the profile with the specified JSON settings. The following is returned, which indicates that the profile is created.

```
{"AuthenticationProfileName": "ExampleProfileName",
"AuthenticationProfileContent": "{\"AllowDBUserOverride\":\"1\",
\"Client_ID\":\"ExampleClientID\",\"App_ID\":\"ExampleAppID\",
\"AutoCreate\":false,\"enableFetchRingBuffer\":true,
\"databaseMetadataCurrentDbOnly\":true}" }
```

Limitations and quotas for creating an authentication profile

Each customer has a quota of ten (10) authentication profiles.

Certain errors can occur with authentication profiles. Examples are if you create a new profile with an existing name, or if you exceed your profile quota. For more information, see CreateAuthenticationProfile.

You can't store certain option keys and values for JDBC, ODBC, and Python connection strings in the authentication profile store:

- `AccessKeyID`
- `access_key_id`
- `SecretAccessKey`
- `secret_access_key_id`
- `PWD`
- `Password`
- `password`

You can't store the key or value `AuthProfile` in the profile store, for JDBC or ODBC connection strings. For Python connections, you can't store `auth_profile`.

Authentication profiles are stored in Amazon DynamoDB and managed by AWS.

Connecting with an authentication profile

After you create an authentication profile, you can include the profile name as a connection option for JDBC version 2.x `AuthProfile`. Using this connection option retrieves the stored settings.

```
jdbc:redshift:iam://endpoint:port/database?AuthProfile=<Profile-
Name>&AccessKeyID=<Caller-Access-Key>&SecretAccessKey=<Caller-Secret-Key>
```

The following is an example JDBC URL string.

```
jdbc:redshift:iam://examplecluster:us-west-2/dev?
AuthProfile="ExampleProfile"&AccessKeyID="AKIAIOSFODNN7EXAMPLE"&SecretAccessKey="wJalrXUtnFEMI/
K7MDENG/bPxRfiCYEXAMPLEKEY"
```

Specify both the `AccessKeyID` and `SecretAccessKey` in the JDBC URL, along with the authentication profile name.

You can also separate the configuration options with semicolon delimiters, such as in the following example, which includes options for logging.

```
jdbc:redshift:iam://my_redshift_end_point:5439/dev?LogLevel=6;LogPath=/
tmp;AuthProfile=my_profile;AccessKeyID="AKIAIOSFODNN7EXAMPLE";SecretAccessKey="wJalrXUtnFEMI/
K7MDENG/bPxRfiCYEXAMPLEKEY"
```

> **Note**
>
> Don't add confidential information to the authentication profile. For example, don't store an `AccessKeyID` or `SecretAccessKey` value in an authentication profile. The authentication profile store has rules to prohibit storage of secret keys. You get an error if you try to store a key and value associated with sensitive information.

Getting authentication profiles

To list existing authentication profiles, call the following command.

```
describe-authentication-profiles --authentication-profile-name <value: String>
```

The following example shows two retrieved profiles. All profiles are returned if you don't specify a profile name.

```
{ "AuthenticationProfiles": [ { "AuthenticationProfileName":
"testProfile1", "AuthenticationProfileContent": "{\"AllowDBUserOverride
\":\"1\",\"Client_ID\":\"ExampleClientID\",\"App_ID\":\"ExampleAppID
\",\"AutoCreate\":false,\"enableFetchRingBuffer\":true,
\"databaseMetadataCurrentDbOnly\":true}" }, { "AuthenticationProfileName":
"testProfile2", "AuthenticationProfileContent": "{\"AllowDBUserOverride
\":\"1\",\"Client_ID\":\"ExampleClientID\",\"App_ID\":\"ExampleAppID
\",\"AutoCreate\":false,\"enableFetchRingBuffer\":true,
\"databaseMetadataCurrentDbOnly\":true}" } ] }
```

Troubleshooting connection issues in Amazon Redshift

If you have issues with connecting to your cluster from a SQL client tool, there are several things that you can check to narrow down the problem. If you are using SSL or server certificates, first

remove this complexity while you troubleshoot the connection issue. Then add this back when you have found a solution. For more information, see Configuring security options for connections.

For information about behavior changes in Amazon Redshift functionality that may affect your application, see Behavior changes in Amazon Redshift.

> **Important**
>
> Amazon Redshift has changed the way that SSL certificates are managed. If you have trouble connecting using SSL, you might need to update your current trust root CA certificates. For more information, see Transitioning to ACM certificates for SSL connections.

The following section has some example error messages and possible solutions for connection issues. Because different SQL client tools provide different error messages, this is not a complete list, but should be a good starting point for troubleshooting issues.

Connecting from outside of Amazon EC2 and encountering a firewall timeout issue

Your client connection to the database appears to hang or timeout when running long queries, such as a COPY command. In this case, you might observe that the Amazon Redshift console displays that the query has completed, but the client tool itself still appears to be running the query. The results of the query might be missing or incomplete depending on when the connection stopped.

Possible solutions

This issue happens when you connect to Amazon Redshift from a machine other than an Amazon EC2 instance. In this case, idle connections are terminated by an intermediate network component, such as a firewall, after a period of inactivity. This behavior is typical when you log on from a virtual private network (VPN) or your local network.

To avoid these timeouts, we recommend the following changes:

- Increase client system values that deal with TCP/IP timeouts. Make these changes on the computer you are using to connect to your cluster. The timeout period should be adjusted for your client and network. For more information, see Change TCP/IP timeout settings.

- Optionally, set keepalive behavior at the DSN level. For more information, see Change DSN timeout settings.

Change TCP/IP timeout settings

To change TCP/IP timeout settings, configure the timeout settings according to the operating system that you use to connect to your cluster.

- Linux — If your client is running on Linux, run the following command as the root user to change the timeout settings for the current session:

  ```
  /sbin/sysctl -w net.ipv4.tcp_keepalive_time=200 net.ipv4.tcp_keepalive_intvl=200
   net.ipv4.tcp_keepalive_probes=5
  ```

  To persist the settings, create or modify the file `/etc/sysctl.conf` with the following values then reboot your system.

  ```
  net.ipv4.tcp_keepalive_time=200
  net.ipv4.tcp_keepalive_intvl=200
  net.ipv4.tcp_keepalive_probes=5
  ```

- Windows — If your client runs on Windows, edit the values for the following registry settings under HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\:
  - KeepAliveTime: 30000
  - KeepAliveInterval: 1000
  - TcpMaxDataRetransmissions: 10

  These settings use the DWORD data type. If they do not exist under the registry path, you can create the settings and specify these recommended values. For more information about editing the Windows registry, refer to Windows documentation.

  After you set these values, restart your computer for the changes to take effect.

- Mac — If your client is running on a Mac, run the following commands to change the timeout settings for the current session:

```
sudo sysctl net.inet.tcp.keepintvl=200000
sudo sysctl net.inet.tcp.keepidle=200000
sudo sysctl net.inet.tcp.keepinit=200000
sudo sysctl net.inet.tcp.always_keepalive=1
```

To persist the settings, create or modify the file `/etc/sysctl.conf` with the following values:

```
net.inet.tcp.keepidle=200000
net.inet.tcp.keepintvl=200000
net.inet.tcp.keepinit=200000
net.inet.tcp.always_keepalive=1
```

Restart your computer, and then run the following commands to verify that the values are set.

```
sysctl net.inet.tcp.keepidle
sysctl net.inet.tcp.keepintvl
sysctl net.inet.tcp.keepinit
sysctl net.inet.tcp.always_keepalive
```

Change DSN timeout settings

You can set keepalive behavior at the DSN level if you choose. You do this by adding or modifying the following parameters in the odbc.ini file:

KeepAlivesCount

The number of TCP keepalive packets that can be lost before the connection is considered broken.

KeepAlivesIdle

The number of seconds of inactivity before the driver sends a TCP keepalive packet.

KeepAlivesInterval

The number of seconds between each TCP keepalive retransmission.

If these parameters don't exist, or if they have a value of 0, the system uses the keepalive parameters specified for TCP/IP to determine DSN keepalive behavior. On Windows, you can find the TCP/IP parameters in the registry in `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet`

\Services\Tcpip\Parameters\. On Linux and macOS, you can find the TCP/IP parameters can be found in the sysctl.conf file.

Connection is refused or fails

When your connection if refused or fails, you may receive an error similar to one of the following.

- "Failed to establish a connection to *<endpoint>*."
- "Could not connect to server: Connection timed out. Is the server running on host *'<endpoint>'* and accepting TCP/IP connections on port *'<port>'*?"
- "Connection refused. Check that the hostname and port are correct and that the postmaster is accepting TCP/IP connections."

Possible solutions

Generally, when you receive an error message indicating that there is a failure to establish a connection, it is an issue with permission to access the cluster or with network traffic reaching the cluster.

To connect to the cluster from a client tool outside of the network that the cluster is in, you add a inbound rule to the cluster's security group. The rule configuration depends on whether the Amazon Redshift cluster is created in a virtual private cloud (VPC):

- If you created the Amazon Redshift cluster in a virtual private cloud (VPC) based on Amazon VPC, add an inbound rule to the VPC security group that specifies the client CIDR/IP address, in Amazon VPC. For more information about configuring the VPC security groups for your cluster and publicly accessible options, see Redshift resources in a VPC.
- If you created your Amazon Redshift cluster outside a VPC, add your client CIDR/IP address to the cluster security group in Amazon Redshift. For more information about configuring cluster security groups, see Amazon Redshift security groups.

If you attempt to connect to the cluster from a client tool that runs on an Amazon EC2 instance, you also add an inbound rule. In this case, add a rule to the cluster security group. The rule must specify the Amazon EC2 security group associated with the client tool's Amazon EC2 instance.

In some cases, you might have a layer between your client and server, such as a firewall. In these cases, make sure that the firewall accepts inbound connections over the port that you configured for your cluster.

Client and driver are incompatible

If your client and driver are incompatible, you may receive an error that says, "The specified DSN contains an architecture mismatch between the Driver and Application."

Possible solutions

When you attempt to connect and get an error about an architecture mismatch, this means that the client tool and the driver aren't compatible. This occurs because their system architecture doesn't match. For example, this can happen if you have a 32-bit client tool but have installed the 64-bit version of the driver. Sometimes 64-bit client tools can use 32-bit drivers, but you can't use 32-bit applications with 64-bit drivers. Make sure that the driver and client tool are using the same version of the system architecture.

Queries appear to hang and sometimes fail to reach the cluster

You experience an issue with queries completing, where the queries appear to be running but hang in the SQL client tool. Sometimes the queries fail to appear in the cluster, such as in system tables or the Amazon Redshift console.

Possible solutions

This issue can happen due to packet drop. In this case, there is a difference in the maximum transmission unit (MTU) size in the network path between two Internet Protocol (IP) hosts. The MTU size determines the maximum size, in bytes, of a packet that can be transferred in one Ethernet frame over a network connection. In AWS, some Amazon EC2 instance types support an MTU of 1500 (Ethernet v2 frames) and other instance types support an MTU of 9001 (TCP/IP jumbo frames).

To avoid issues that can occur with differences in MTU size, we recommend doing one of the following:

- If your cluster uses the EC2-VPC platform, configure the Amazon VPC security group with an inbound custom Internet Control Message Protocol (ICMP) rule that returns `Destination Unreachable`. The rule thus instructs the originating host to use the lowest MTU size along the network path. For details on this approach, see Configuring security groups to allow ICMP "destination unreachable".
- If your cluster uses the EC2-Classic platform, or you can't allow the ICMP inbound rule, disable TCP/IP jumbo frames so that Ethernet v2 frames are used. For details on this approach, see Configuring the MTU of an instance.

Configuring security groups to allow ICMP "destination unreachable"

When there is a difference in the MTU size in the network between two hosts, first make sure that your network settings don't block path MTU discovery (PMTUD). PMTUD enables the receiving host to respond to the originating host with the following ICMP message: `Destination Unreachable: fragmentation needed and DF set (ICMP Type 3, Code 4)`. This message instructs the originating host to use the lowest MTU size along the network path to resend the request. Without this negotiation, packet drop can occur because the request is too large for the receiving host to accept. For more information about this ICMP message, go to RFC792 on the *Internet Engineering Task Force (IETF)* website.

If you don't explicitly configure this ICMP inbound rule for your Amazon VPC security group, PMTUD is blocked. In AWS, security groups are virtual firewalls that specify rules for inbound and outbound traffic to an instance. For information about Amazon Redshift cluster security group, see Amazon Redshift security groups. For clusters using the EC2-VPC platform, Amazon Redshift uses VPC security groups to allow or deny traffic to the cluster. By default, the security groups are locked down and deny all inbound traffic. For information about how to set inbound and outbound rules for EC2-Classic or EC2-VPC instances, see Differences between instances in EC2-Classic and a VPC in the *Amazon EC2 User Guide*.

For more information about how to add rules to VPC security groups, see VPC security groups. For more information about specific PMTUD settings required in this rule, see Path MTU discovery in the *Amazon EC2 User Guide.*

Configuring the MTU of an instance

In some cases, your cluster might use the EC2-Classic platform or you can't allow the custom ICMP rule for inbound traffic. In these cases, we recommend that you adjust the MTU to 1500 on the network interface (NIC) of the EC2 instances you connect to your Amazon Redshift cluster from. This adjustment disables TCP/IP jumbo frames to ensure that connections consistently use the same packet size. However, this option reduces your maximum network throughput for the instance entirely, not just for connections to Amazon Redshift. For more information, see the following procedures.

To set MTU on a Microsoft Windows operating system

If your client runs in a Microsoft Windows operating system, you can review and set the MTU value for the Ethernet adapter by using the `netsh` command.

1. Run the following command to determine the current MTU value:

```
netsh interface ipv4 show subinterfaces
```

2. Review the MTU value for the Ethernet adapter in the output.
3. If the value is not 1500, run the following command to set it:

```
netsh interface ipv4 set subinterface "Ethernet" mtu=1500 store=persistent
```

After you set this value, restart your computer for the changes to take effect.

To set MTU on a Linux operating system

If your client runs in a Linux operating system, you can review and set the MTU value by using the ip command.

1. Run the following command to determine the current MTU value:

```
$ ip link show eth0
```

2. Review the value following mtu in the output.
3. If the value is not 1500, run the following command to set it:

```
$ sudo ip link set dev eth0 mtu 1500
```

To set MTU on a Mac operating system

- Follow instructions on the macOS support site about How to change the MTU for troubleshooting purposes. For more information, search the support site.

Setting the JDBC fetch size parameter

By default, the Redshift JDBC driver uses a ring buffer to manage memory efficiently and prevent out-of-memory errors. The fetch size parameter is only applicable when the ring buffer is explicitly disabled. For more information, review the link. In this configuration, you should set the fetch size to control how many rows are retrieved in each batch.

When to Use Fetch Size?

Use the fetch size parameter when:

- You need fine-grained control over row-based batching
- Working with legacy applications that require traditional fetch size behavior

Setting Fetch Size

When ring buffer is disabled, the JDBC driver collects all results for a query at one time by default. Queries that return large result sets can consume excessive memory. To retrieve result sets in batches instead of all at once, set the JDBC fetch size parameter in your application.

> **Note**
>
> Fetch size is not supported for ODBC.

For the best performance, set the fetch size to the highest value that does not lead to out of memory errors. A lower fetch size value results in more server trips, which prolong execution times. The server reserves resources, including the WLM query slot and associated memory, until the client retrieves the entire result set or the query is canceled. When you tune the fetch size appropriately, those resources are released more quickly, making them available to other queries.

> **Note**
>
> If you need to extract large datasets, we recommend using an UNLOAD statement to transfer the data to Amazon S3. When you use UNLOAD, the compute nodes work in parallel to speed up the transfer of data.

For more information about setting the JDBC fetch size parameter, go to Getting results based on a cursor in the PostgreSQL documentation.

Using the Amazon Redshift Data API

The Amazon Redshift Data API simplifies access to your Amazon Redshift data warehouse by removing the need to manage database drivers, connections, network configurations, data

buffering, credentials, and more. You can run SQL statements using the Data API operations with the AWS SDK. For more information about the Data API operations, see the Amazon Redshift Data API Reference.

The Data API doesn't require a persistent connection to your database. Instead, it provides a secure HTTP endpoint and integration with AWS SDKs. You can use the endpoint to run SQL statements without managing connections. Calls to the Data API are asynchronous. The Data API can use either credentials stored in AWS Secrets Manager or temporary database credentials. You don't need to pass passwords in the API calls with either authorization method. For more information about AWS Secrets Manager, see What Is AWS Secrets Manager? in the *AWS Secrets Manager User Guide*. You can also use AWS IAM Identity Center for authorization.

With the Data API, you can programmatically access Amazon Redshift data with web services–based applications, including AWS Lambda, Amazon SageMaker AI notebooks, and AWS Cloud9. For more information on these applications, see AWS Lambda, Amazon SageMaker AI, and AWS Cloud9.

To learn more about the Data API, see Get started with the Amazon Redshift Data API in the *AWS Big Data Blog*.

Working with the Amazon Redshift Data API

Before you use the Amazon Redshift Data API, review the following steps:

1. Determine if you, as the caller of the Data API, are authorized. For more information about authorization, see Authorizing access to the Amazon Redshift Data API.
2. Determine if you plan to call the Data API with authentication credentials from Secrets Manager, temporary credentials, or use AWS IAM Identity Center. For more information, see Choosing database authentication credentials when calling the Amazon Redshift Data API.
3. Set up a secret if you use Secrets Manager for authentication credentials. For more information, see Storing database credentials in AWS Secrets Manager.
4. Review the considerations and limitations when calling the Data API. For more information, see Considerations when calling the Amazon Redshift Data API.
5. Call the Data API from the AWS Command Line Interface (AWS CLI), from your own code, or using the query editor in the Amazon Redshift console. For examples of calling from the AWS CLI, see Calling the Data API.

Considerations when calling the Amazon Redshift Data API

Consider the following when calling the Data API:

- The Amazon Redshift Data API can access databases in Amazon Redshift provisioned clusters and Redshift Serverless workgroups. For a list of AWS Regions where the Redshift Data API is available, see the endpoints listed for Redshift Data API in the *Amazon Web Services General Reference*.
- The maximum duration of a query is 24 hours.
- The maximum number of active queries (`STARTED` and `SUBMITTED` queries) per Amazon Redshift cluster is 500.
- The maximum query result size is 500 MB (after gzip compression). If a call returns more than 500 MB of response data, the call is ended.
- The maximum retention time for query results is 24 hours.
- The maximum query statement size is 200 KB.
- The Data API is available to query all Serverless workgroups, and Provisioned single-node and multiple-node clusters of the following node types:
  - dc2.large
  - dc2.8xlarge
  - rg.large
  - rg.xlarge
  - rg.4xlarge
  - rg.12xlarge
  - ra3.large
  - ra3.xlplus
  - ra3.4xlarge
  - ra3.16xlarge
- The cluster must be in a virtual private cloud (VPC) based on the Amazon VPC service.
- By default, users with the same IAM role as the runner of an `ExecuteStatement` or `BatchExecuteStatement` API operation can act on the same statement with `CancelStatement`, `DescribeStatement`, `GetStatementResult`, `GetStatementResultV2`, and `ListStatements` API operations. To act on the same SQL statement from another user, the user must be able to assume the IAM role of the user who ran the SQL statement. For more

information about how to assume a role, see Authorizing access to the Amazon Redshift Data API.

- The SQL statements in the `Sqls` parameter of the `BatchExecuteStatement` API operation run serially in the order of the array. Subsequent SQL statements don't start until the previous statement completes. By default, all SQL statements are run as a single transaction. If any SQL statement fails, all work is rolled back. Use the `ExecutionMode` parameter to control transaction behavior:
  - `TRANSACTION` (default) — The service runs all SQL statements as a single transaction and commits or rolls back all of them together.
  - `AUTO_COMMIT` — Each SQL statement is committed individually. A failure of one statement does not affect the others.

  For an example, see Run multiple SQL statements.
- The maximum retention time for a client token used in `ExecuteStatement` or `BatchExecuteStatement` API operation is 8 hours.
- The optional `WaitTimeSeconds` parameter has the Data API wait up to 30 seconds when the operation submits a new query or is against an in-progress query. This reduces the number of poll requests, resulting in fewer round trips and lower latency. For more information, see Reduce API calls with long polling.
- If the Amazon Redshift provisioned clusters and Redshift Serverless workgroup is encrypted using a customer managed key Redshift creates a grant that allows the Redshift Data API to use the key for its operations. For for more information, see Using AWS KMS with the Amazon Redshift Data API.
- Each API in the Redshift Data API has a transactions per second quota before throttling requests. For the quota, see Quotas for Amazon Redshift Data API. If the rate of request exceeds the quota, a `ThrottlingException` with HTTP Status Code: 400 is returned. To respond to throttling, use a retry strategy as described in Retry behavior in the *AWS SDKs and Tools Reference Guide*. This strategy is implemented automatically for throttling errors in some AWS SDKs.

> **Note**
>
> By default in AWS Step Functions, retries are not enabled. If you need to call a Redshift Data API in a Step Functions state machine, then include the `ClientToken` idempotency parameter in your Redshift Data API call. The value of the `ClientToken` needs to persist among retries. In the following example snippet of a request to the `ExecuteStatement`

API, the expression `States.ArrayGetItem(States.StringSplit($$.Execution.Id, ':'), 7)` uses an intrinsic function to extract the UUID part of the `$$.Execution.Id`, which is unique for each execution of the state machine. For more information, see Intrinsic functions in the *AWS Step Functions Developer Guide*.

```
{
  "Database": "dev",
  "Sql": "select 1;",
  "ClusterIdentifier": "MyCluster",
  "ClientToken.$": "States.ArrayGetItem(States.StringSplit($$.Execution.Id,
 ':'), 7)"
}
```

Choosing database authentication credentials when calling the Amazon Redshift Data API

When you call the Data API, you use one of the following authentication methods for some API operations. Each method requires a different combination of parameters.

AWS IAM Identity Center

The Data API can be accessed with a single sign-on user registered in AWS IAM Identity Center. For information about the steps to set up IAM Identity Center, see Using Data API with trusted identity propagation.

AWS Secrets Manager

With this method, provide the `secret-arn` of a secret stored in AWS Secrets Manager which has `username` and `password`. The specified secret contains credentials to connect to the `database` you specify. When you are connecting to a cluster, you also supply the database name, If you provide a cluster identifier (dbClusterIdentifier), it must match the cluster identifier stored in the secret. When you are connecting to a serverless workgroup, you also supply the database name. For more information, see Storing database credentials in AWS Secrets Manager.

With this method, you can also supply a `region` value that specifies the AWS Region where your data is located.

Temporary credentials

With this method, choose one of the following options:

- When connecting to a serverless workgroup, specify the workgroup name and database name. The database user name is derived from the IAM identity. For example, `arn:iam::123456789012:user:foo` has the database user name `IAM:foo`. Also, permission to call the `redshift-serverless:GetCredentials` operation is required.
- When connecting to a cluster as an IAM identity, specify the cluster identifier and the database name. The database user name is derived from the IAM identity. For example, `arn:iam::123456789012:user:foo` has the database user name `IAM:foo`. Also, permission to call the `redshift:GetClusterCredentialsWithIAM` operation is required.
- When connecting to a cluster as a database user, specify the cluster identifier, the database name, and the database user name. Also, permission to call the `redshift:GetClusterCredentials` operation is required. For information about how to join database groups when connecting with this method, see Joining database groups when connecting to a cluster.

With this method, you can also supply a `region` value that specifies the AWS Region where your data is located.

Mapping JDBC data types when calling the Amazon Redshift Data API

The following table maps Java Database Connectivity (JDBC) data types to the data types you specify in Data API calls.

JDBC data type	Data API data type
`INTEGER, SMALLINT, BIGINT`	`LONG`
`FLOAT, REAL, DOUBLE`	`DOUBLE`
`DECIMAL`	`STRING`
`BOOLEAN, BIT`	`BOOLEAN`
`BLOB, BINARY, LONGVARBINARY`	`BLOB`
`VARBINARY`	`STRING`

JDBC data type	Data API data type
CLOB	STRING
Other types (including types related to date and time)	STRING

String values are passed to the Amazon Redshift database and implicitly converted into a database data type.

> **Note**
>
> Currently, the Data API doesn't support arrays of universal unique identifiers (UUIDs).

Running SQL statements with parameters when calling the Amazon Redshift Data API

You can control the SQL text submitted to the database engine by calling the Data API operation using parameters for parts of the SQL statement. Named parameters provide a flexible way to pass in parameters without hardcoding them in the SQL text. They help you reuse SQL text and avoid SQL injection problems.

The following example shows the named parameters of a `parameters` field of an `execute-statement` or `batch-execute-statement` AWS CLI command.

```
--parameters "[{\"name\": \"id\", \"value\": \"1\"},{\"name\": \"address\", \"value\":
 \"Seattle\"}]"
```

Consider the following when using named parameters:

- Named parameters can only be used to replace values in SQL statements.
  - You can replace the values in an INSERT statement, such as `INSERT INTO mytable VALUES(:val1)`.

    The named parameters can be in any order and parameters can be used more than one time in the SQL text. The parameters option shown in a previous example, the values 1 and `Seattle`

are inserted into the table columns `id` and `address`. In the SQL text, you specify the named parameters as follows:

```
--sql "insert into mytable values (:id, :address)"
```

- You can replace the values in a conditions clause, such as `WHERE attr >= :val1`, `WHERE attr BETWEEN :val1 AND :val2`, and `HAVING COUNT(attr) > :val`.
- You can't replace column names in an SQL statement, such as `SELECT column-name`, `ORDER BY column-name`, or `GROUP BY column-name`.

  For example, the following SELECT statement fails with invalid syntax.

  ```
  --sql "SELECT :colname, FROM event" --parameters "[{\"name\": \"colname\", \"value
  \": \"eventname\"}]"
  ```

  If you describe (`describe-statement` operation) the statement with the syntax error, the `QueryString` returned does not substitute the column name for the parameter (`"QueryString": "SELECT :colname, FROM event"`), and an error is reported (ERROR: syntax error at or near \"FROM\"\n Position: 12).
- You can't replace column names in an aggregate function, such as `COUNT(column-name)`, `AVG(column-name)`, or `SUM(column-name)`.
- You can't replace column names in a JOIN clause.

- When the SQL runs, data is implicitly cast to a data type. For more information about data type casting, see Data types in the *Amazon Redshift Database Developer Guide*.
- You can't set a value to NULL. The Data API interprets it as the literal string `NULL`. The following example replaces `id` with the literal string `null`. Not the SQL NULL value.

  ```
  --parameters "[{\"name\": \"id\", \"value\": \"null\"}]"
  ```
- You can't set a zero length value. The Data API SQL statement fails. The following example trys to set `id` with a zero length value and results in a failure of the SQL statement.

  ```
  --parameters "[{\"name\": \"id\", \"value\": \"\"}]"
  ```
- You can't set a table name in the SQL statement with a parameter. The Data API follows the rule of the JDBC `PreparedStatement`.

- The output of the `describe-statement` operation returns the query parameters of a SQL statement.
- Both the `execute-statement` and `batch-execute-statement` operations support SQL statements with parameters. When using `batch-execute-statement`, parameters are shared across all SQL statements in the batch. Each SQL statement can reference a subset of the provided parameters, but every parameter must be used by at least one SQL statement.

Running SQL statements with an idempotency token when calling the Amazon Redshift Data API

When you make a mutating API request, the request typically returns a result before the operation's asynchronous workflows have completed. Operations might also time out or encounter other server issues before they complete, even though the request has already returned a result. This could make it difficult to determine whether the request succeeded or not, and could lead to multiple retries to ensure that the operation completes successfully. However, if the original request and the subsequent retries are successful, the operation is completed multiple times. This means that you might update more resources than you intended.

Idempotency ensures that an API request completes no more than one time. With an idempotent request, if the original request completes successfully, any subsequent retries complete successfully without performing any further actions. The Data API `ExecuteStatement` and `BatchExecuteStatement` operations have an optional `ClientToken` idempotent parameter. The `ClientToken` expires after 8 hours.

> **Important**
>
> If you call `ExecuteStatement` and `BatchExecuteStatement` operations from an AWS SDK, it automatically generates a client token to use on retry. In this case, we don't recommend using the `client-token` parameter with `ExecuteStatement` and `BatchExecuteStatement` operations. View the CloudTrail log to see the `ClientToken`. For a CloudTrail log example, see Amazon Redshift Data API examples.

The following `execute-statement` AWS CLI command illustrates the optional `client-token` parameter for idempotency.

```
aws redshift-data execute-statement
    --secret-arn arn:aws:secretsmanager:us-west-2:123456789012:secret:myuser-secret-
hKgPWn
    --cluster-identifier mycluster-test
    --sql "select * from stl_query limit 1"
    --database dev
    --client-token b855dced-259b-444c-bc7b-d3e8e33f94g1
```

The following table shows some common responses that you might get for idempotent API requests, and provides retry recommendations.

Response	Recommendation	Comments
200 (OK)	Do not retry	The original request completed successfully. Any subsequent retries return successfully.
400-series response codes	Do not retry	There is a problem with the request, from among the following: • It includes a parameter or parameter combination that is not valid. • It uses an action or resource for which you do not have permissions. • It uses a resource that is in the process of changing states. If the request involves a resource that is in the process of changing states, retrying the request could possibly succeed.
500-series response codes	Retry	The error is caused by an AWS server-side issue and is generally transient. Repeat the request with an appropriate backoff strategy.

For information about Amazon Redshift response codes, see Common Errors in the *Amazon Redshift API Reference*.

Running SQL statements with session reuse when calling the Amazon Redshift Data API

When you make an API request to run a SQL statement, the session where the SQL runs is usually terminated when the SQL is finished. To keep the session active for a specified number of seconds, the Data API `ExecuteStatement` and `BatchExecuteStatement` operations have an optional `SessionKeepAliveSeconds` parameter. A `SessionId` response field contains the identity of the session which can then be used in subsequent `ExecuteStatement` and `BatchExecuteStatement` operations. In subsequent calls you can specify another `SessionKeepAliveSeconds` to change the idle timeout time. If the `SessionKeepAliveSeconds` is not changed, the initial idle timeout setting remains. Consider the following when using session reuse:

- The maximum value of `SessionKeepAliveSeconds` is 24 hours.
- The session can last for at most 24 hours. After 24 hours the session is forcibly closed and in-progress queries are terminated.
- The maximum number of sessions per Amazon Redshift cluster or Redshift Serverless workgroup is 500.
- You can only run one query at a time in a session. You need to wait until the query is finished to run the next query in the same session. That is, you cannot run queries in parallel in a provided session.
- The Data API can't queue queries for a given session.

To retrieve the `SessionId` that is used by calls to `ExecuteStatement` and `BatchExecuteStatement` operations, call `DescribeStatement` and `ListStatements` operations. To list sessions and their status, use the `ListSessions` operation. For more information, see List sessions.

The following example demonstrates using the `SessionKeepAliveSeconds` and `SessionId` parameters to keep a session alive and reused. First, call the `execute-statement` AWS CLI command with the optional `session-keep-alive-seconds` parameter set to 2.

```
aws redshift-data execute-statement
    --session-keep-alive-seconds 2
    --sql "select 1"
    --database dev
```

```
    --workgroup-name mywg
```

The response contains the session identifier.

```
{
    "WorkgroupName": "mywg",
    "CreatedAt": 1703022996.436,
    "Database": "dev",
    "DbUser": "awsuser",
    "Id": "07c5ffea-76d6-4786-b62c-4fe3ef529680",
    "SessionId": "5a254dc6-4fc2-4203-87a8-551155432ee4"
}
```

Then, call the `execute-statement` AWS CLI command with the `SessionId` returned from the first call. And optionally, specify the `session-keep-alive-seconds` parameter set to `10` to change the idle timeout value.

```
aws redshift-data execute-statement
    --sql "select 1"
    --session-id 5a254dc6-4fc2-4203-87a8-551155432ee4
    --session-keep-alive-seconds 10
```

Fetching the results of SQL statements

You use different Data API operations to fetch SQL results depending on the result format. When you call `ExecuteStatement` and `BatchExecuteStatement` operations, you can specify whether the results are formatted as JSON or CSV. If you don't specify, the default is JSON. To retrieve JSON results, use the `GetStatementResult` operation. To retrieve CSV results, use the `GetStatementResultV2` operation.

Results returned in JSON format are records that include metadata about each column. Each record is in JSON format. For example, the response from `GetStatementResult` looks similar to this:

```
{
   "ColumnMetadata": [
      {
         "isCaseSensitive": false,
         "isCurrency": false,
         "isSigned": true,
         "label": "?column?",
```

```
          "name": "?column?",
          "nullable": 1,
          "precision": 10,
          "scale": 0,
          "schemaName": "",
          "tableName": "",
          "typeName": "int4",
          "length": 0
       }
   ],
   "NextToken": "<token>",
   "Records": [
        [
            {
                "longValue": 1
            }
        ]
    ],
   "TotalNumRows": <number>
}
```

Results returned in CSV format are records that include metadata about each column. Results are returned in 1 MB chunks, where each chunk can store any number of rows in CSV format. Each request returns up to 15 MB of results. If results are greater than 15 MB, then a next page token is returned to continue retrieving the results. For example, the response from `GetStatementResultV2` looks similar to this:

```
{
    "ColumnMetadata": [
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "?column?",
            "name": "?column?",
            "nullable": 1,
            "precision": 10,
            "scale": 0,
            "schemaName": "",
            "tableName": "",
            "typeName": "int4",
            "length": 0
        },
```

```
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "?column?",
            "name": "?column?",
            "nullable": 1,
            "precision": 10,
            "scale": 0,
            "schemaName": "",
            "tableName": "",
            "typeName": "int4",
            "length": 0
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "?column?",
            "name": "?column?",
            "nullable": 1,
            "precision": 10,
            "scale": 0,
            "schemaName": "",
            "tableName": "",
            "typeName": "int4",
            "length": 0
        }
    ],
    "NextToken": "<token>",
    "Records": [
        [
            {
                "CSVRecords":"1,2,3\r\n4,5,6\r\n7,8,9\r\n, ... 1MB" // First 1MB Chunk
            },
            {
                "CSVRecords":"1025,1026,1027\r\n1028,1029,1030\r\n...2MB" // Second 1MB
chunk
            }
            ...
        ]
    ],
    "ResultFormat" : "CSV",
    "TotalNumRows": <number>
```

```
}
```

Authorizing access to the Amazon Redshift Data API

To access the Data API, a user must be authorized. You can authorize a user to access the Data API by adding a managed policy, which is a predefined AWS Identity and Access Management (IAM) policy, to that user. As a best practice, we recommend attaching permissions policies to an IAM role and then assigning it to users and groups as needed. For more information, see Identity and access management in Amazon Redshift. To see the permissions allowed and denied by managed policies, see the IAM console (https://console.aws.amazon.com/iam/).

Configuring IAM permissions

Amazon Redshift provides the `AmazonRedshiftDataFullAccess` managed policy. This policy provides full access to Amazon Redshift Data API operations. This policy also allows scoped access to specific Amazon Redshift, AWS Secrets Manager, and IAM API operations needed to authenticate and access an Amazon Redshift cluster or Redshift Serverless workgroup.

You can also create your own IAM policy that allows access to specific resources. To create your policy, use the `AmazonRedshiftDataFullAccess` policy as your starting template. After you create your policy, add it to each user that requires access to the Data API.

Consider the following requirements of the IAM policy associated with the user:

- If you use AWS Secrets Manager to authenticate, confirm the policy allows use of the `secretsmanager:GetSecretValue` action to retrieve the secret tagged with the key `RedshiftDataFullAccess`.
- If you use temporary credentials to authenticate to a cluster, confirm the policy allows the use of the `redshift:GetClusterCredentials` action to the database user name `redshift_data_api_user` for any database in the cluster. This user name must have already been created in your database.
- If you use temporary credentials to authenticate to a serverless workgroup, confirm the policy allows the use of the `redshift-serverless:GetCredentials` action to retrieve the workgroup tagged with the key `RedshiftDataFullAccess`. The database user is mapped 1:1 to the source AWS Identity and Access Management (IAM) identity. For example, the user sample_user is mapped to database user `IAM:sample_user`, and IAM role sample_role is mapped to `IAMR:sample_role`. For more information about IAM identities, see IAM Identities (users, user groups, and roles) in the IAM User Guide.

- The IAM action `redshift-data:GetStatementResult` allows access to both `GetStatementResult` and `GetStatementResultV2` API operations.

The following links provide more information about AWS Identity and Access Management in the *IAM User Guide*.

- For information about creating an IAM roles, see Creating IAM roles.
- For information about creating an IAM policy, see Creating IAM policies.
- For information about adding an IAM policy to a user, see Adding and removing IAM identity permissions.

Run a query on a cluster that is owned by another account

To run a query on a cluster that is owned by another account, the owning account must provide an IAM role that the Data API can assume in the calling account. For example, suppose Account B owns a cluster that Account A needs to access. Account B can attach the AWS managed policy `AmazonRedshiftDataFullAccess` to Account B's IAM role. Then Account B trusts Account A using a trust policy such as the following:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:role/someRoleA"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Finally, the Account A IAM role needs to be able to assume the Account B IAM role.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "arn:aws:iam::111122223333:role/someRoleB"
    }
}
```

Specify an IAM role that restricts resources to Redshift Serverless workgroups and Amazon Redshift clusters in an AWS account

You can specify resource ARNs in your identity-based policy to control access to Redshift Serverless workgroups and Amazon Redshift clusters in an AWS account. This example shows how you might create a policy that allows access to the Data API for only the workgroup and clusters in the specified AWS account.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "redshift-data:CancelStatement",
                "redshift-data:DescribeStatement",
                "redshift-data:GetStatementResult",
                "redshift-data:ListStatements"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "redshift-data:*",
            "Resource": [
                "arn:aws:redshift:us-east-1:111122223333:workgroup/*",
                "arn:aws:redshift:us-east-1:111122223333:cluster:*"
```

```
            ]
        }
    ]
}
```

Configure an IAM policy that restricts access to SQL statement information to only the statement owner

By default, Amazon Redshift Data API treats the IAM role used when calling `ExecuteStatement` and `BatchExecuteStatement` as the owner of the SQL statement. Anyone who is allowed to assume the role is able to access information about the SQL statement, including its results. To restrict SQL statement information access to an IAM role session with a particular owner, add condition `redshift-data:statement-owner-iam-userid: "${aws:userid}"`. The following IAM policy restricts access.

JSON

```
{
"Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "redshift-data:CancelStatement",
                "redshift-data:DescribeStatement",
                "redshift-data:GetStatementResult",
                "redshift-data:ListStatements"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "redshift-data:statement-owner-iam-userid": "${aws:userid}"
                }
            }
        }
    ]
}
```

You can use the condition `statement-owner-iam-userid` with `CancelStatement`, `DescribeStatement`, `GetStatementResult`, and `ListStatements`. For more information, see Actions defined by Amazon Redshift Data API.

Configure an IAM policy that restricts access to SQL results to only the session owner

By default, Amazon Redshift Data API treats the IAM role used when calling `ExecuteStatement` and `BatchExecuteStatement` as the owner of the database session that runs the SQL statement. Anyone who is allowed to assume the role is able to submit queries to the database session. To restrict session access to an IAM role session with a particular owner, add condition `redshift-data:session-owner-iam-userid: "${aws:userid}"`. The following IAM policy restricts access.

The following IAM policy allows only the session owner to get statement results. The condition `session-owner-iam-userid` is used to limit resource access to the specified `userid`.

JSON

```
{
"Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "redshift-data:ExecuteStatement",
                "redshift-data:BatchExecuteStatement"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "redshift-data:session-owner-iam-userid": "${aws:userid}"
                }
            }
        }
    ]
}
```

You can use the condition `session-owner-iam-userid` with `ExecuteStatement` and `BatchExecuteStatement`. For more information, see Actions defined by Amazon Redshift Data API.

Storing database credentials in AWS Secrets Manager

When you call the Data API, you can pass credentials for the cluster or serverless workgroup by using a secret in AWS Secrets Manager. To pass credentials in this way, you specify the name of the secret or the Amazon Resource Name (ARN) of the secret.

To store credentials with Secrets Manager, you need `SecretManagerReadWrite` managed policy permission. For more information about the minimum permissions, see Creating and Managing Secrets with AWS Secrets Manager in the *AWS Secrets Manager User Guide*.

To store your credentials in a secret for an Amazon Redshift cluster

1. Use the AWS Secrets Manager console to create a secret that contains credentials for your cluster:

   - When you choose **Store a new secret**, choose **Credentials for Redshift cluster**.
   - Store your values for **User name** (database user), **Password**, and **DB cluster** (cluster identifier) in your secret.
   - Tag the secret with the key `RedshiftDataFullAccess`. The AWS managed policy `AmazonRedshiftDataFullAccess` only allows the action `secretsmanager:GetSecretValue` for secrets tagged with the key `RedshiftDataFullAccess`.

   For instructions, see Creating a Basic Secret in the *AWS Secrets Manager User Guide*.
2. Use the AWS Secrets Manager console to view the details for the secret you created, or run the `aws secretsmanager describe-secret` AWS CLI command.

   Note the name and ARN of the secret. You can use these in calls to the Data API.

To store your credentials in a secret for a serverless workgroup

1. Use AWS Secrets Manager AWS CLI commands to store a secret that contains credentials for your serverless workgroup:

   - Create your secret in a file, for example a JSON file named `mycreds.json`. Provide the values for **User name** (database user) and **Password** in the file.

     ```
     {
     ```

```
      "username": "myusername",
      "password": "mypassword"
}
```

- Store your values in your secret and tag the secret with the key `RedshiftDataFullAccess`.

```
aws secretsmanager create-secret --name MyRedshiftSecret  --tags
 Key="RedshiftDataFullAccess",Value="serverless" --secret-string file://
mycreds.json
```

The following shows the output.

```
{
    "ARN":
 "arn:aws:secretsmanager:region:accountId:secret:MyRedshiftSecret-mvLHxf",
    "Name": "MyRedshiftSecret",
    "VersionId": "a1603925-e8ea-4739-9ae9-e509eEXAMPLE"
}
```

For more information, see Creating a Basic Secret with AWS CLI in the *AWS Secrets Manager User Guide*.

2. Use the AWS Secrets Manager console to view the details for the secret you created, or run the `aws secretsmanager describe-secret` AWS CLI command.

   Note the name and ARN of the secret. You can use these in calls to the Data API.

Creating an Amazon VPC endpoint (AWS PrivateLink) for the Data API

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources, such as Amazon Redshift clusters and applications, into a virtual private cloud (VPC). AWS PrivateLink provides private connectivity between virtual private clouds (VPCs) and AWS services securely on the Amazon network. Using AWS PrivateLink, you can create VPC endpoints, which you can use connect to services across different accounts and VPCs based on Amazon VPC. For more information about AWS PrivateLink, see VPC Endpoint Services (AWS PrivateLink) in the *Amazon Virtual Private Cloud User Guide*.

You can call the Data API with Amazon VPC endpoints. Using an Amazon VPC endpoint keeps traffic between applications in your Amazon VPC and the Data API in the AWS network, without using public IP addresses. Amazon VPC endpoints can help you meet compliance and regulatory requirements related to limiting public internet connectivity. For example, if you use an Amazon VPC endpoint, you can keep traffic between an application running on an Amazon EC2 instance and the Data API in the VPCs that contain them.

After you create the Amazon VPC endpoint, you can start using it without making any code or configuration changes in your application.

To create an Amazon VPC endpoint for the Data API

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Choose **Endpoints**, and then choose **Create Endpoint**.
3. On the **Create Endpoint** page, for **Service category**, choose **AWS services**. For **Service Name**, choose **redshift-data** (`com.amazonaws.region.redshift-data`).
4. For **VPC**, choose the VPC to create the endpoint in.

   Choose the VPC that contains the application that makes Data API calls.
5. For **Subnets**, choose the subnet for each Availability Zone (AZ) used by the AWS service that is running your application.

   To create an Amazon VPC endpoint, specify the private IP address range in which the endpoint is accessible. To do this, choose the subnet for each Availability Zone. Doing so restricts the VPC endpoint to the private IP address range specific to each Availability Zone and also creates an Amazon VPC endpoint in each Availability Zone.
6. For **Enable DNS name**, select **Enable for this endpoint**.

   Private DNS resolves the standard Data API DNS hostname (`https://redshift-data.region.amazonaws.com`) to the private IP addresses associated with the DNS hostname specific to your Amazon VPC endpoint. As a result, you can access the Data API VPC endpoint using the AWS CLI or AWS SDKs without making any code or configuration changes to update the Data API endpoint URL.
7. For **Security group**, choose a security group to associate with the Amazon VPC endpoint.

   Choose the security group that allows access to the AWS service that is running your application. For example, if an Amazon EC2 instance is running your application, choose the

security group that allows access to the Amazon EC2 instance. The security group enables you to control the traffic to the Amazon VPC endpoint from resources in your VPC.

8. Choose **Create endpoint**.

After the endpoint is created, choose the link in the AWS Management Console to view the endpoint details.

The endpoint **Details** tab shows the DNS hostnames that were generated while creating the Amazon VPC endpoint.

You can use the standard endpoint (`redshift-data.region.amazonaws.com`) or one of the VPC-specific endpoints to call the Data API within the Amazon VPC. The standard Data API endpoint automatically routes to the Amazon VPC endpoint. This routing occurs because the Private DNS hostname was enabled when the Amazon VPC endpoint was created.

When you use an Amazon VPC endpoint in a Data API call, all traffic between your application and the Data API remains in the Amazon VPCs that contain them. You can use an Amazon VPC endpoint for any type of Data API call. For information about calling the Data API, see Considerations when calling the Amazon Redshift Data API.

Joining database groups when connecting to a cluster

Database groups are collections of database users. Database privileges can be granted to groups. An administrator can configure an IAM role such that these database groups are taken into account when your SQL runs with the Data API. For more information about database groups, see Groups in the *Amazon Redshift Database Developer Guide*.

You can configure a Data API caller's IAM role so that the database user specified in the call joins database groups when the Data API connects to a cluster. This capability is only supported when connecting to provisioned clusters. It's not supported when connecting to Redshift Serverless workgroups. The IAM role of the caller of the Data API must also allow the `redshift:JoinGroup` action.

Configure this by adding tags to IAM roles. The administrator of the caller's IAM role adds tags with the key `RedshiftDbGroups` and a key value of a list of database groups. The value is a list of colon (:) separated names of database groups up to a total length of 256 characters. The database groups must be previously defined in the connected database. If any specified group is not found in the database, it's ignored. For example, for database groups `accounting` and `retail`, the key-value is `accounting:retail`. The tag key-value pair

`{"Key":"RedshiftDbGroups","Value":"accounting:retail"}` is used by the Data API to determine which database groups are associated with the provided database user in the call to the Data API.

To join database groups

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane of the console, choose **Roles** and then choose the name of the role that you want to edit.
3. Choose the **Tags** tab, then choose **Manage tags**.
4. Choose **Add tag**, then add the key **RedshiftDbGroups** and a value which is a list of *database-groups-colon-separated*.
5. Choose **Save changes**.

   Now when an IAM principal (with this IAM role attached) calls the Data API, the specified database user joins the database groups specified in the IAM role.

For more information on how to attach a tag to a principal, including IAM roles and IAM users, see Tagging IAM resources in the *IAM User Guide*.

Using Data API with trusted identity propagation

As an Amazon Redshift account administrator, you can integrate your Amazon Redshift cluster or workgroup with AWS IAM Identity Center, which helps manage your workforce access to Amazon Redshift with single sign-on. For more information, see Setting up AWS IAM Identity Center integration with Amazon Redshift. The Amazon Redshift Data API supports propagating IAM Identity Center user identities to an Amazon Redshift cluster or workgroup, and to other services, such as, AWS Lake Formation, down the chain. You can set up and query using the Data API by following the steps in Access AWS services programmatically using trusted identity propagation.

When you call the Data API using an IAM Identity Center user identity from an identity-enhanced IAM role session, you can only access the resulting statement and statement result using the same IAM Identity Center user. For example, the following AWS CLI command calls the `execute-statement` operation to run a SQL command with trusted identity propagation.

```
aws redshift-data execute-statement
--sql "select current_user;"
```

```
--cluster-id mycluster
--database dev
```

The following AWS CLI command calls the `batch-execute-statement` operation to run two SQL commands.

```
aws redshift-data batch-execute-statement
--sqls  "select current_user;"  "select current_date;"
--cluster-id mycluster
--database dev
```

To access statements with `cancel-statement`, `describe-statement`, `get-statement-result`, and `get-statement-result-v2` submitted by identity-enhanced IAM role sessions, the IAM Identity Center user and IAM role must match the credentials used to run `execute-statement` or `batch-execute-statement`. For example, the following AWS CLI command gets the results of a SQL statement.

```
aws redshift-data get-statement-result
--id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

To list statements, a `cluster-identifier` or `workgroup-name` parameter must be provided to ensure that the IAM Identity Center user only has access the Amazon Redshift IAM Identity Center applications they are assigned to. For example, the following AWS CLI command lists statements for a specific cluster.

```
aws redshift-data list-statements
--cluster-identifier mycluster
```

You can also invoke the Data API operations that access database objects in a cluster or workgroup using trusted identity propagation. This includes the `list-databases`, `list-schemas`, `list-tables`, and `describe-table` operations.

API calls made by the IAM Identity Center user can be tracked in AWS CloudTrail. A `onBehalfOf` section of the CloudTrail event shows the IAM Identity Center user id and the identity store ARN. The following example shows a snippet of a CloudTrail event showing the `onBehalfOf` section with the IAM Identity Center user ID of `a1b2c3d4-5678-90ab-cdef-EXAMPLE11111` and the Identity store ARN of `arn:aws:identitystore::123456789012:identitystore/d-9067bc44d2`.

```
{
            "eventVersion":"1.10",
            "userIdentity":{
            "type":"AssumedRole",
            ...
            },
            "onBehalfOf":{
            "userId":"a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "identityStoreArn":"arn:aws:identitystore::123456789012:identitystore/
d-9067bc44d2"
            }
            },
            "eventTime":"2025-01-13T04:46:27Z",
            "eventSource":"redshift-data.amazonaws.com",
            "eventName":"ExecuteStatement",
            "awsRegion":"us-east-1"
            }
```

You can run the following SQL command to check the query submitted by the IAM Identity Center user. In this example, the email registered in Identity Center is `username@example.com`.

```
SELECT
    h.query_id,
    h.database_name,
    h.status,
    h.query_text,
    u.usename,
    h.start_time,
    h.end_time
FROM
    sys_query_history h
LEFT JOIN
    pg_user u
ON
    h.user_id = u.usesysid
where u.usename='awsidc:username@example.com'
ORDER BY
    h.start_time DESC;
```

Calling the Data API

You can call the Data API or the AWS CLI to run SQL statements on your cluster or serverless workgroup. The primary operations to run SQL statements are `ExecuteStatement` and `BatchExecuteStatement` in the *Amazon Redshift Data API Reference*. The Data API supports the programming languages that are supported by the AWS SDK. For more information on these, see Tools to Build on AWS.

To see code examples of calling the Data API, see Getting Started with Redshift Data API in *GitHub*. This repository has examples of using AWS Lambda to access Amazon Redshift data from Amazon EC2, AWS Glue Data Catalog, and Amazon SageMaker Runtime. Example programming languages include Python, Go, Java, and Javascript.

You can call the Data API using the AWS CLI.

The following examples use the AWS CLI to call the Data API. To run the examples, edit the parameter values to match your environment. In many of the examples a `cluster-identifier` is provided to run against a cluster. When you run against a serverless workgroup, you provide a `workgroup-name` instead. These examples demonstrate a few of the Data API operations. For more information, see the *AWS CLI Command Reference*.

Commands in the following examples have been split and formatted for readability. Not all parameters and responses are shown in all examples. For the API definition of the complete request syntax, request parameters, response syntax, and response elements, see the Amazon Redshift Data API Reference.

Passing SQL statements to an Amazon Redshift data warehouse

The examples in this page cover different ways to pass a SQL statement to your data warehouse

Run a SQL statement

To run a SQL statement, use the `aws redshift-data execute-statement` AWS CLI command.

Optionally, use the `WaitTimeSeconds` parameter to allow extra time for the statement to complete before the API returns a response. When the statement completes, the response also includes `Status`, `RedshiftPid`, and `HasResultSet`. For more information, see Reduce API calls with long polling.

The following AWS CLI command runs a SQL statement against a cluster and returns an identifier to fetch the results. This example uses the AWS Secrets Manager authentication method.

```
aws redshift-data execute-statement
    --secret-arn arn:aws:secretsmanager:us-west-2:123456789012:secret:myuser-secret-
hKgPWn
    --cluster-identifier mycluster-test
    --sql "select * from stl_query limit 1"
    --database dev
```

The following is an example of the response.

```
{
    "ClusterIdentifier": "mycluster-test",
    "CreatedAt": 1598323175.823,
    "Database": "dev",
    "Id": "c016234e-5c6c-4bc5-bb16-2c5b8ff61814",
    "SecretArn": "arn:aws:secretsmanager:us-west-2:123456789012:secret:myuser-secret-
hKgPWn"
}
```

The following AWS CLI command runs a SQL statement against a cluster and returns an identifier to fetch the results. This example uses the temporary credentials authentication method.

```
aws redshift-data execute-statement
    --db-user myuser
    --cluster-identifier mycluster-test
    --database dev
    --sql "select * from stl_query limit 1"
```

The following is an example of the response.

```
{
    "ClusterIdentifier": "mycluster-test",
    "CreatedAt": 1598306924.632,
    "Database": "dev",
    "DbUser": "myuser",
    "Id": "d9b6c0c9-0747-4bf4-b142-e8883122f766"
}
```

The following AWS CLI command runs a SQL statement against a serverless workgroup and returns an identifier to fetch the results. This example uses the temporary credentials authentication method.

```
aws redshift-data execute-statement
    --database dev
    --workgroup-name myworkgroup
    --sql "select 1;"
```

The following is an example of the response.

```
{
 "CreatedAt": "2022-02-11T06:25:28.748000+00:00",
 "Database": "dev",
 "DbUser": "IAMR:RoleName",
 "Id": "89dd91f5-2d43-43d3-8461-f33aa093c41e",
 "WorkgroupName": "myworkgroup"
}
```

The following AWS CLI command runs a SQL statement against a cluster and returns an identifier to fetch the results. This example uses the AWS Secrets Manager authentication method and an idempotency token.

```
aws redshift-data execute-statement
    --secret-arn arn:aws:secretsmanager:us-west-2:123456789012:secret:myuser-secret-
hKgPWn
    --cluster-identifier mycluster-test
    --sql "select * from stl_query limit 1"
    --database dev
    --client-token b855dced-259b-444c-bc7b-d3e8e33f94g1
```

The following is an example of the response.

```
{
    "ClusterIdentifier": "mycluster-test",
    "CreatedAt": 1598323175.823,
    "Database": "dev",
    "Id": "c016234e-5c6c-4bc5-bb16-2c5b8ff61814",
```

```
    "SecretArn": "arn:aws:secretsmanager:us-west-2:123456789012:secret:myuser-secret-
hKgPWn"
}
```

Run multiple SQL statements

To run multiple SQL statements with one command, use the `aws redshift-data batch-execute-statement` AWS CLI command.

Optionally, use the `WaitTimeSeconds` parameter to allow extra time for the entire batch to complete before the API returns a response. For more information, see Reduce API calls with long polling.

The following AWS CLI command runs three SQL statements against a cluster and returns an identifier to fetch the results. This example uses the temporary credentials authentication method.

```
aws redshift-data batch-execute-statement
    --db-user myuser
    --cluster-identifier mycluster-test
    --database dev
    --sqls "set timezone to BST" "select * from mytable" "select * from another_table"
```

The following is an example of the response.

```
{
    "ClusterIdentifier": "mycluster-test",
    "CreatedAt": 1598306924.632,
    "Database": "dev",
    "DbUser": "myuser",
    "Id": "d9b6c0c9-0747-4bf4-b142-e8883122f766"
}
```

The following AWS CLI command runs three SQL statements in separate transactions using `AUTO_COMMIT` execution mode. Each statement is committed independently — a failure of one statement does not stop the remaining statements from running or affect statements that already completed.

```
aws redshift-data batch-execute-statement
    --db-user myuser
```

```
    --cluster-identifier mycluster-test
    --database dev
    --sqls "insert into mytable values (1)" "invalid sql statement" "insert into
 mytable values (3)"
    --execution-mode AUTO_COMMIT
```

The following is an example of the response.


```
{
    "ClusterIdentifier": "mycluster-test",
    "CreatedAt": 1623979777.126,
    "Duration": 6591877,
    "HasResultSet": false,
    "Id": "b2906c76-fa6e-4cdf-8c5f-4de1ff9b7652",
    "RedshiftPid": 31459,
    "RedshiftQueryId": 0,
    "Status": "FAILED",
    "SubStatements": [
        {
            "CreatedAt": 1623979777.274,
            "Duration": 3396637,
            "HasResultSet": false,
            "Id": "b2906c76-fa6e-4cdf-8c5f-4de1ff9b7652:1",
            "QueryString": "insert into mytable values (1)",
            "RedshiftQueryId": -1,
            "Status": "FINISHED",
            "UpdatedAt": 1623979777.903
        },
        {
            "CreatedAt": 1623979777.274,
            "Duration": 1195240,
            "HasResultSet": false,
            "Id": "b2906c76-fa6e-4cdf-8c5f-4de1ff9b7652:2",
            "QueryString": "invalid sql statement",
            "RedshiftQueryId": -1,
            "Error": "ERROR: syntax error at or near \"invalid\"",
            "Status": "FAILED",
            "UpdatedAt": 1623979777.503
        },
        {
            "CreatedAt": 1623979777.274,
            "Duration": 1095981,
            "HasResultSet": false,
```

```
            "Id": "b2906c76-fa6e-4cdf-8c5f-4de1ff9b7652:3",
            "QueryString": "insert into mytable values (3)",
            "RedshiftQueryId": -1,
            "Status": "FINISHED",
            "UpdatedAt": 1623979777.903
        }
    ],
    "UpdatedAt": 1623979778.183
}
```

Run statements with parameters

You can use named parameters with both the `execute-statement` and `batch-execute-statement` operations. When using `batch-execute-statement`, parameters are shared across all SQL statements in the batch. Each SQL statement can reference a subset of the provided parameters, but every parameter must be used by at least one SQL statement.

The following AWS CLI command runs a SQL statement against a cluster and returns an identifier to fetch the results. This example uses the AWS Secrets Manager authentication method. The SQL text has named parameter `distance`. In this case, the distance used in the predicate is 5. In a SELECT statement, named parameters for column names can only be used in the predicate. Values for named parameters for the SQL statement are specified in the `parameters` option.

```
aws redshift-data execute-statement
    --secret-arn arn:aws:secretsmanager:us-west-2:123456789012:secret:myuser-secret-
hKgPWn
    --cluster-identifier mycluster-test
    --sql "SELECT ratecode FROM demo_table WHERE trip_distance > :distance"
    --parameters "[{\"name\": \"distance\", \"value\": \"5\"}]"
    --database dev
```

The following is an example of the response.

```
{
    "ClusterIdentifier": "mycluster-test",
    "CreatedAt": 1598323175.823,
    "Database": "dev",
    "Id": "c016234e-5c6c-4bc5-bb16-2c5b8ff61814",
    "SecretArn": "arn:aws:secretsmanager:us-west-2:123456789012:secret:myuser-secret-
hKgPWn"
```

```
}
```

The following example uses the EVENT table from the sample database. For more information, see EVENT table in the *Amazon Redshift Database Developer Guide*.

If you don't already have the EVENT table in your database, you can create one using the Data API as follows:

```
aws redshift-data execute-statement
--database dev
--cluster-id mycluster-test
--db-user awsuser
--sql "create table event( eventid integer not null distkey,
                           venueid smallint not null,
                           catid smallint not null,
                           dateid smallint not null sortkey,
                           eventname varchar(200),
                           starttime timestamp)"
```

The following command inserts one row into the EVENT table.

```
aws redshift-data execute-statement
--database dev
--cluster-id mycluster-test
--db-user awsuser
--sql "insert into event
 values(:eventid, :venueid::smallint, :catid, :dateid, :eventname, :starttime)"
--parameters "[{\"name\": \"eventid\", \"value\": \"1\"}, {\"name\": \"venueid\",
 \"value\": \"1\"},
               {\"name\": \"catid\", \"value\": \"1\"},
               {\"name\": \"dateid\", \"value\": \"1\"},
               {\"name\": \"eventname\", \"value\": \"event 1\"},
               {\"name\": \"starttime\", \"value\": \"2022-02-22\"}]"
```

The following command inserts a second row into the EVENT table. This example demonstrates the following:

- The parameter named `id` is used four times in the SQL text.
- Implicit type conversion is applied automatically when inserting parameter `starttime`.

- The `venueid` column is type cast to SMALLINT data type.
- Character strings that represent the DATE data type are implicitly converted into the TIMESTAMP data type.
- Comments can be used within SQL text.

```
aws redshift-data execute-statement
--database dev
--cluster-id mycluster-test
--db-user awsuser
--sql "insert into event values(:id, :id::smallint, :id, :id, :eventname, :starttime) /
*this is comment, and it won't apply parameterization for :id, :eventname or :starttime
 here*/"
--parameters "[{\"name\": \"eventname\", \"value\": \"event 2\"},
               {\"name\": \"starttime\", \"value\": \"2022-02-22\"},
               {\"name\": \"id\", \"value\": \"2\"}]"
```

The following shows the two inserted rows:

```
 eventid | venueid | catid | dateid | eventname |      starttime
---------+---------+-------+--------+-----------+---------------------
       1 |       1 |     1 |      1 | event 1   | 2022-02-22 00:00:00
       2 |       2 |     2 |      2 | event 2   | 2022-02-22 00:00:00
```

The following command uses a named parameter in a WHERE clause to retrieve the row where `eventid` is 1.

```
aws redshift-data execute-statement
--database dev
--cluster-id mycluster-test
--db-user awsuser
--sql "select * from event where eventid=:id"
--parameters "[{\"name\": \"id\", \"value\": \"1\"}]"
```

Run the following command to get the SQL results of the previous SQL statement:

```
aws redshift-data get-statement-result --id 7529ad05-b905-4d71-9ec6-8b333836eb5a
```

Provides the following results:


```
{
    "Records": [
        [
            {
                "longValue": 1
            },
            {
                "longValue": 1
            },
            {
                "longValue": 1
            },
            {
                "longValue": 1
            },
            {
                "stringValue": "event 1"
            },
            {
                "stringValue": "2022-02-22 00:00:00.0"
            }
        ]
    ],
    "ColumnMetadata": [
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "eventid",
            "length": 0,
            "name": "eventid",
            "nullable": 0,
            "precision": 10,
            "scale": 0,
            "schemaName": "public",
            "tableName": "event",
```

```
            "typeName": "int4"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "venueid",
            "length": 0,
            "name": "venueid",
            "nullable": 0,
            "precision": 5,
            "scale": 0,
            "schemaName": "public",
            "tableName": "event",
            "typeName": "int2"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "catid",
            "length": 0,
            "name": "catid",
            "nullable": 0,
            "precision": 5,
            "scale": 0,
            "schemaName": "public",
            "tableName": "event",
            "typeName": "int2"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "dateid",
            "length": 0,
            "name": "dateid",
            "nullable": 0,
            "precision": 5,
            "scale": 0,
            "schemaName": "public",
            "tableName": "event",
            "typeName": "int2"
        },
```

```
        {
            "isCaseSensitive": true,
            "isCurrency": false,
            "isSigned": false,
            "label": "eventname",
            "length": 0,
            "name": "eventname",
            "nullable": 1,
            "precision": 200,
            "scale": 0,
            "schemaName": "public",
            "tableName": "event",
            "typeName": "varchar"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": false,
            "label": "starttime",
            "length": 0,
            "name": "starttime",
            "nullable": 1,
            "precision": 29,
            "scale": 6,
            "schemaName": "public",
            "tableName": "event",
            "typeName": "timestamp"
        }
    ],
    "TotalNumRows": 1
}
```

The following AWS CLI command runs two SQL statements with shared named parameters against a cluster using `batch-execute-statement`. The SQL text has named parameters `id` and `name`. The first SQL statement uses the `id` parameter and the second uses the `name` parameter.

```
aws redshift-data batch-execute-statement
    --db-user myuser
    --cluster-identifier mycluster-test
    --database dev
    --sqls "SELECT * FROM mytable WHERE id = :id" "SELECT * FROM mytable WHERE name
 = :name"
```

```
    --parameters "[{\"name\": \"id\", \"value\": \"1\"},{\"name\": \"name\", \"value\":
 \"Alice\"}]"
```

The following is an example of the response.

```
{
    "ClusterIdentifier": "mycluster-test",
    "CreatedAt": 1598306924.632,
    "Database": "dev",
    "DbUser": "myuser",
    "Id": "d9b6c0c9-0747-4bf4-b142-e8883122f766"
}
```

The following AWS CLI command runs two SQL statements where the same parameter is used across both statements. The parameter named `id` is used in an INSERT statement and a SELECT statement.

```
aws redshift-data batch-execute-statement
    --db-user myuser
    --cluster-identifier mycluster-test
    --database dev
    --sqls "INSERT INTO mytable VALUES(:id, :name, :address)" "SELECT * FROM mytable
 WHERE id = :id"
    --parameters "[{\"name\": \"id\", \"value\": \"1\"},{\"name\": \"name\", \"value\":
 \"Alice\"},{\"name\": \"address\", \"value\": \"Seattle\"}]"
```

The following is an example of the response.

```
{
    "ClusterIdentifier": "mycluster-test",
    "CreatedAt": 1598306924.632,
    "Database": "dev",
    "DbUser": "myuser",
    "Id": "d9b6c0c9-0747-4bf4-b142-e8883122f766"
}
```

List metadata about SQL statements

To list metadata about SQL statements, use the `aws redshift-data list-statements` AWS CLI command. Authorization to run this command is based on the caller's IAM permissions.

The following AWS CLI command lists SQL statements that ran.

```
aws redshift-data list-statements
    --status ALL
```

The following is an example of the response.

```
{
    "Statements": [
        {
            "CreatedAt": 1598306924.632,
            "Id": "d9b6c0c9-0747-4bf4-b142-e8883122f766",
            "QueryString": "select * from stl_query limit 1",
            "Status": "FINISHED",
            "UpdatedAt": 1598306926.667
        },
        {
            "CreatedAt": 1598311717.437,
            "Id": "e0ebd578-58b3-46cc-8e52-8163fd7e01aa",
            "QueryString": "select * from stl_query limit 1",
            "Status": "FAILED",
            "UpdatedAt": 1598311719.008
        },
        {
            "CreatedAt": 1598313683.65,
            "Id": "c361d4f7-8c53-4343-8c45-6b2b1166330c",
            "QueryString": "select * from stl_query limit 1",
            "Status": "ABORTED",
            "UpdatedAt": 1598313685.495
        },
        {
            "CreatedAt": 1598306653.333,
            "Id": "a512b7bd-98c7-45d5-985b-a715f3cfde7f",
            "QueryString": "select 1",
            "Status": "FINISHED",
            "UpdatedAt": 1598306653.992
        }
    ]
}
```

List sessions

To list the sessions that you created in the last 24 hours, use the `aws redshift-data list-sessions` AWS CLI command. By default, the command returns only sessions with a status of `AVAILABLE` or `BUSY`. You can filter the results by session status, compute target (cluster or serverless workgroup), or database. Your IAM permissions determine whether you can run this command. For more information about sessions, see Running SQL statements with session reuse when calling the Amazon Redshift Data API.

The following AWS CLI command lists the sessions that ran on a cluster:

```
aws redshift-data list-sessions
   --cluster-identifier mycluster-test
```

The following is an example of the response:

```
{
    "Sessions": [
        {
            "SessionId": "5a254dc6-4fc2-4203-87a8-551155432ee4",
            "Status": "AVAILABLE",
            "CreatedAt": 1703022996.436,
            "UpdatedAt": 1703023010.221,
            "Database": "dev",
            "DbUser": "awsuser",
            "ClusterIdentifier": "mycluster-test",
            "SessionAliveSeconds": 10,
            "SessionTtl": 1703023020.436
        },
        {
            "SessionId": "8b365ed7-5gd7-5314-98b9-662266543ff5",
            "Status": "BUSY",
            "CreatedAt": 1703023100.512,
            "UpdatedAt": 1703023150.874,
            "Database": "dev",
            "DbUser": "awsuser",
            "ClusterIdentifier": "mycluster-test",
            "SessionAliveSeconds": 60,
            "SessionTtl": 1703023210.512,
            "CurrentStatementId": "c016234e-5c6c-4bc5-bb16-2c5b8ff61814"
        }
```

```
    ]
}
```

The following AWS CLI command lists the CLOSED sessions on a serverless workgroup by filtering on session status:

```
aws redshift-data list-sessions
    --workgroup-name myworkgroup
    --status CLOSED
```

The following is an example of the response:

```
{
    "Sessions": [
        {
            "SessionId": "07c5ffea-76d6-4786-b62c-4fe3ef529680",
            "Status": "CLOSED",
            "CreatedAt": 1703019996.436,
            "UpdatedAt": 1703020056.436,
            "Database": "dev",
            "DbUser": "IAMR:RoleName",
            "WorkgroupName": "myworkgroup",
            "SessionAliveSeconds": 60,
            "SessionTtl": 1703020056.436
        }
    ]
}
```

Note

If you use identity-enhanced IAM role sessions, then you must provide either the `cluster-identifier` or `workgroup-name` parameter. With this parameter, you ensure that the AWS IAM Identity Center user can access only the Amazon Redshift IAM Identity Center applications that they are assigned to. For more information, see Using Data API with trusted identity propagation.

Describe metadata about a SQL statement

To get descriptions of metadata for a SQL statement, use the `aws redshift-data describe-statement` AWS CLI command. You can also use this command to describe individual sub-statements within a batch by providing the sub-statement ID. Authorization to run this command is based on the caller's IAM permissions.

Optionally, to reduce the number of poll requests, use the `WaitTimeSeconds` parameter to give extra time for an operation's status to be updated within this API call. For more information, see Reduce API calls with long polling.

The following AWS CLI command describes a SQL statement.

```
aws redshift-data describe-statement
    --id d9b6c0c9-0747-4bf4-b142-e8883122f766
```

The following is an example of the response.

```
{
    "ClusterIdentifier": "mycluster-test",
    "CreatedAt": 1598306924.632,
    "Duration": 1095981511,
    "Id": "d9b6c0c9-0747-4bf4-b142-e8883122f766",
    "QueryString": "select * from stl_query limit 1",
    "RedshiftPid": 20859,
    "RedshiftQueryId": 48879,
    "ResultRows": 1,
    "ResultSize": 4489,
    "Status": "FINISHED",
    "UpdatedAt": 1598306926.667
}
```

The following is an example of a `describe-statement` response after running a `batch-execute-statement` command with multiple SQL statements.

```
{
    "ClusterIdentifier": "mayo",
    "CreatedAt": 1623979777.126,
    "Duration": 6591877,
```

```
    "HasResultSet": true,
    "Id": "b2906c76-fa6e-4cdf-8c5f-4de1ff9b7652",
    "RedshiftPid": 31459,
    "RedshiftQueryId": 0,
    "ResultRows": 2,
    "ResultSize": 22,
    "Status": "FINISHED",
    "SubStatements": [
        {
            "CreatedAt": 1623979777.274,
            "Duration": 3396637,
            "HasResultSet": true,
            "Id": "b2906c76-fa6e-4cdf-8c5f-4de1ff9b7652:1",
            "QueryString": "select 1;",
            "RedshiftQueryId": -1,
            "ResultRows": 1,
            "ResultSize": 11,
            "Status": "FINISHED",
            "UpdatedAt": 1623979777.903
        },
        {
            "CreatedAt": 1623979777.274,
            "Duration": 3195240,
            "HasResultSet": true,
            "Id": "b2906c76-fa6e-4cdf-8c5f-4de1ff9b7652:2",
            "QueryString": "select 2;",
            "RedshiftQueryId": -1,
            "ResultRows": 1,
            "ResultSize": 11,
            "Status": "FINISHED",
            "UpdatedAt": 1623979778.076
        }
    ],
    "UpdatedAt": 1623979778.183
}
```

The following AWS CLI command describes a specific sub-statement within a batch by providing the sub-statement ID.

```
aws redshift-data describe-statement
    --id b2906c76-fa6e-4cdf-8c5f-4de1ff9b7652:2
```

The following is an example of the response. The parent statement status is STARTED because the batch has not yet completed. The SubStatements array contains only the requested sub-statement.


```
{
    "ClusterIdentifier": "mayo",
    "CreatedAt": 1623979777.126,
    "Duration": 0,
    "HasResultSet": false,
    "Id": "b2906c76-fa6e-4cdf-8c5f-4de1ff9b7652",
    "RedshiftPid": 31459,
    "RedshiftQueryId": 0,
    "Status": "STARTED",
    "SubStatements": [
        {
            "CreatedAt": 1623979777.274,
            "Duration": 3195240,
            "HasResultSet": true,
            "Id": "b2906c76-fa6e-4cdf-8c5f-4de1ff9b7652:2",
            "QueryString": "select 2;",
            "RedshiftQueryId": -1,
            "ResultRows": 1,
            "ResultSize": 11,
            "Status": "FINISHED",
            "UpdatedAt": 1623979778.076
        }
    ],
    "UpdatedAt": 1623979777.503
}
```


Fetch the results of a SQL statement

To fetch the result from a SQL statement that ran, use the `redshift-data get-statement-result` or `redshift-data get-statement-result-v2` AWS CLI command. The results from `get-statement-result` are in JSON format. The results from `get-statement-result-v2` are in CSV format. You can provide an `Id` that you receive in response to `execute-statement` or `batch-execute-statement`. The `Id` value for a SQL statement run by `batch-execute-statement` can be retrieved in the result of `describe-statement` and is suffixed by a colon and sequence number such as `b2906c76-fa6e-4cdf-8c5f-4de1ff9b7652:2`. If you run multiple SQL statements with `batch-execute-statement`, each SQL statement has an `Id` value as

shown in `describe-statement`. Authorization to run this command is based on the caller's IAM permissions.

Optionally, to reduce the number of poll requests, use the `WaitTimeSeconds` parameter to allow extra time for the statement to complete before the API returns a statement result. For more information, see Reduce API calls with long polling.

The following statement returns the result of a SQL statement run by `execute-statement` that let the `ResultFormat` default to `JSON`. To retrieve the results, call the `get-statement-result` operation.

```
aws redshift-data get-statement-result
    --id d9b6c0c9-0747-4bf4-b142-e8883122f766
```

The following statement returns the result of the second SQL statement run by `batch-execute-statement`.

```
aws redshift-data get-statement-result
    --id b2906c76-fa6e-4cdf-8c5f-4de1ff9b7652:2
```

The following is an example of the response to a call to `get-statement-result` where the SQL result is returned in JSON format in the `Records` key of the response.

```
{
    "ColumnMetadata": [
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "userid",
            "length": 0,
            "name": "userid",
            "nullable": 0,
            "precision": 10,
            "scale": 0,
            "schemaName": "",
            "tableName": "stll_query",
            "typeName": "int4"
        },
```

```
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "query",
            "length": 0,
            "name": "query",
            "nullable": 0,
            "precision": 10,
            "scale": 0,
            "schemaName": "",
            "tableName": "stll_query",
            "typeName": "int4"
        },
        {
            "isCaseSensitive": true,
            "isCurrency": false,
            "isSigned": false,
            "label": "label",
            "length": 0,
            "name": "label",
            "nullable": 0,
            "precision": 320,
            "scale": 0,
            "schemaName": "",
            "tableName": "stll_query",
            "typeName": "bpchar"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "xid",
            "length": 0,
            "name": "xid",
            "nullable": 0,
            "precision": 19,
            "scale": 0,
            "schemaName": "",
            "tableName": "stll_query",
            "typeName": "int8"
        },
        {
            "isCaseSensitive": false,
```

```
            "isCurrency": false,
            "isSigned": true,
            "label": "pid",
            "length": 0,
            "name": "pid",
            "nullable": 0,
            "precision": 10,
            "scale": 0,
            "schemaName": "",
            "tableName": "stll_query",
            "typeName": "int4"
        },
        {
            "isCaseSensitive": true,
            "isCurrency": false,
            "isSigned": false,
            "label": "database",
            "length": 0,
            "name": "database",
            "nullable": 0,
            "precision": 32,
            "scale": 0,
            "schemaName": "",
            "tableName": "stll_query",
            "typeName": "bpchar"
        },
        {
            "isCaseSensitive": true,
            "isCurrency": false,
            "isSigned": false,
            "label": "querytxt",
            "length": 0,
            "name": "querytxt",
            "nullable": 0,
            "precision": 4000,
            "scale": 0,
            "schemaName": "",
            "tableName": "stll_query",
            "typeName": "bpchar"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": false,
```

```
            "label": "starttime",
            "length": 0,
            "name": "starttime",
            "nullable": 0,
            "precision": 29,
            "scale": 6,
            "schemaName": "",
            "tableName": "stll_query",
            "typeName": "timestamp"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": false,
            "label": "endtime",
            "length": 0,
            "name": "endtime",
            "nullable": 0,
            "precision": 29,
            "scale": 6,
            "schemaName": "",
            "tableName": "stll_query",
            "type": 93,
            "typeName": "timestamp"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "aborted",
            "length": 0,
            "name": "aborted",
            "nullable": 0,
            "precision": 10,
            "scale": 0,
            "schemaName": "",
            "tableName": "stll_query",
            "typeName": "int4"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "insert_pristine",
```

```
            "length": 0,
            "name": "insert_pristine",
            "nullable": 0,
            "precision": 10,
            "scale": 0,
            "schemaName": "",
            "tableName": "stll_query",
            "typeName": "int4"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "concurrency_scaling_status",
            "length": 0,
            "name": "concurrency_scaling_status",
            "nullable": 0,
            "precision": 10,
            "scale": 0,
            "schemaName": "",
            "tableName": "stll_query",
            "typeName": "int4"
        }
    ],
    "Records": [
        [
            {
                "longValue": 1
            },
            {
                "longValue": 3
            },
            {
                "stringValue": "health"
            },
            {
                "longValue": 1023
            },
            {
                "longValue": 15279
            },
            {
                "stringValue": "dev"
            },
```

```
            {
                "stringValue": "select system_status from stv_gui_status;"
            },
            {
                "stringValue": "2020-08-21 17:33:51.88712"
            },
            {
                "stringValue": "2020-08-21 17:33:52.974306"
            },
            {
                "longValue": 0
            },
            {
                "longValue": 0
            },
            {
                "longValue": 6
            }
        ]
    ],
    "TotalNumRows": 1
}
```

The following example shows a SQL statement run by `execute-statement` to return results as JSON. The table `testingtable` has three integer columns (col1, col2, col3) and there are three rows with values (1, 2, 3), (4, 5, 6), and (7, 8, 9).

```
aws redshift-data execute-statement
    --database dev
    --sql "SELECT col1, col2, col3 FROM testingtable"
    --cluster-id mycluster-test
    --result-format JSON
```

```
{
    "ClusterIdentifier": "mycluster-test",
    "CreatedAt": "2024-04-02T16:45:25.144000+00:00",
    "Database": "dev",
    "DbUser": "IAMR:Administrator",
    "Id": "d468d942-6df9-4f85-8ae3-bac01a61aec3"
}
```

The following is an example of the response to a call to `get-statement-result` where the SQL result is returned in JSON format in the `Records` key of the response.

```
aws redshift-data get-statement-result
    --id d468d942-6df9-4f85-8ae3-bac01a61aec3
```

```
{
    "Records": [
        [
            {
                "longValue": 1
            },
            {
                "longValue": 2
            },
            {
                "longValue": 3
            }
        ],
        [
            {
                "longValue": 4
            },
            {
                "longValue": 5
            },
            {
                "longValue": 6
            }
        ],
        [
            {
                "longValue": 7
            },
            {
                "longValue": 8
            },
            {
                "longValue": 9
            }
        ]
    ],
```

```
    "ColumnMetadata": [
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "col1",
            "name": "col1",
            "nullable": 1,
            "precision": 10,
            "scale": 0,
            "schemaName": "public",
            "tableName": "testingtable",
            "typeName": "int4",
            "length": 0
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "col2",
            "name": "col2",
            "nullable": 1,
            "precision": 10,
            "scale": 0,
            "schemaName": "public",
            "tableName": "testingtable",
            "typeName": "int4",
            "length": 0
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "col3",
            "name": "col3",
            "nullable": 1,
            "precision": 10,
            "scale": 0,
            "schemaName": "public",
            "tableName": "testingtable",
            "typeName": "int4",
            "length": 0
        }
    ],
```

```
    "TotalNumRows": 3
}
```

The following example shows a SQL statement run by `execute-statement` to return results as a CSV. The table `testingtable` has three integer columns (col1, col2, col3) and there are three rows with values (1, 2, 3), (4, 5, 6), and (7, 8, 9).

```
aws redshift-data execute-statement
    --database dev
    --sql "SELECT col1, col2, col3 FROM testingtable"
    --cluster-id mycluster-test
    --result-format CSV
```

```
{
    "ClusterIdentifier": "mycluster-test",
    "CreatedAt": "2024-04-02T16:45:25.144000+00:00",
    "Database": "dev",
    "DbUser": "IAMR:Administrator",
    "Id": "d468d942-6df9-4f85-8ae3-bac01a61aec3"
}
```

The following is an example of the response to a call to `get-statement-result-v2` where the SQL result is returned in CSV format in the `Records` key of the response. Rows are separated by carriage return and newline (\r\n). The first row returned in `Records` are the column headers. Results returned in CSV format are returned in 1 MB where each chunk can store any number of rows up to 1MB.

```
aws redshift-data get-statement-result-v2
    --id d468d942-6df9-4f85-8ae3-bac01a61aec3
```

```
{
    "Records": [
        {
            "CSVRecords": "col1,col2,col3\r\n1,2,3\r\n4,5,6\r\n7,8,9\r\n"
        }
    ],
    "ColumnMetadata": [
        {
```

```
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "col1",
            "name": "col1",
            "nullable": 1,
            "precision": 10,
            "scale": 0,
            "schemaName": "public",
            "tableName": "testingtable",
            "typeName": "int4",
            "length": 0
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "col2",
            "name": "col2",
            "nullable": 1,
            "precision": 10,
            "scale": 0,
            "schemaName": "public",
            "tableName": "testingtable",
            "typeName": "int4",
            "length": 0
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": true,
            "label": "col3",
            "name": "col3",
            "nullable": 1,
            "precision": 10,
            "scale": 0,
            "schemaName": "public",
            "tableName": "testingtable",
            "typeName": "int4",
            "length": 0
        }
    ],
    "TotalNumRows": 3,
    "ResultFormat": "csv"
```

```
}
```

Reduce API calls with long polling

By default, Data API operations return an immediate response. If the operation is against an in-progress statement, you must poll repeatedly until the statement reaches a terminal state. With long polling, set `WaitTimeSeconds` to a value between 1 and 30. The API will delay returning a synchronous response until the statement finishes, or `WaitTimeSeconds` is reached, whichever comes sooner.

Long polling is supported on DescribeStatement, ExecuteStatement, BatchExecuteStatement, GetStatementResult, and GetStatementResultV2.

When an operation uses `WaitTimeSeconds` to submit a new statement or wait on an existing one:

- If the statement **completes** within the wait time, the response includes the terminal status or result data, depending on the operation.
- If the statement is **still running** when the wait time expires, the response returns the current in-progress status.
- If the statement **already completed** before the long-poll request is made, the API returns immediately with the terminal status.
- If `WaitTimeSeconds` is **omitted**, the default asynchronous behavior is preserved.

Common use cases

The following are examples of when long polling may be useful:

- **Submit a short query and wait for completion** — Use `ExecuteStatement` with `WaitTimeSeconds`. When the query finishes or the wait time expires, the response returns the statement `Status`, `RedshiftPid`, and `HasResultSet`. To get full statement metadata, call `DescribeStatement`. If `HasResultSet` is `true`, call `GetStatementResult` to fetch results.
- **Long poll for status** — Use `ExecuteStatement` to submit the query, then call `DescribeStatement` with `WaitTimeSeconds`. When the query finishes, `DescribeStatement` returns full statement metadata including status, duration, and error details.
- **Long poll for result** — Use `ExecuteStatement` to submit the query, then call `GetStatementResult` with `WaitTimeSeconds`. When the query finishes, result data is returned directly. If the query is still running when the wait time expires, a `ResourceNotFoundException` is returned indicating no results are available yet.

- **Submit multiple queries and wait for the entire batch to complete** — Use `BatchExecuteStatement` with `WaitTimeSeconds`. The call holds open until all sub-statements complete and returns the batch parent ID and overall status.
- **Submit multiple queries and long poll status for a sub-statement** — Use `DescribeStatement` or `GetStatementResult` with the sub-statement ID and `WaitTimeSeconds`. Returns as soon as that sub-statement finishes, without waiting for the rest of the batch.

Describe a table

To get metadata that describes a table, use the `aws redshift-data describe-table` AWS CLI command.

The following AWS CLI command runs a SQL statement against a cluster and returns metadata that describes a table. This example uses the AWS Secrets Manager authentication method.

```
aws redshift-data describe-table
    --cluster-identifier mycluster-test
    --database dev
    --schema information_schema
    --table sql_features
    --secret-arn arn:aws:secretsmanager:us-west-2:123456789012:secret:myuser-secret-
hKgPWn
```

The following is an example of the response.

```
{
    "ColumnList": [
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": false,
            "length": 2147483647,
            "name": "feature_id",
            "nullable": 1,
            "precision": 2147483647,
            "scale": 0,
            "schemaName": "information_schema",
            "tableName": "sql_features",
            "typeName": "character_data"
```

```
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": false,
            "length": 2147483647,
            "name": "feature_name",
            "nullable": 1,
            "precision": 2147483647,
            "scale": 0,
            "schemaName": "information_schema",
            "tableName": "sql_features",
            "typeName": "character_data"
        }
    ]
}
```

The following AWS CLI command runs a SQL statement against a cluster that describes a table. This example uses the temporary credentials authentication method.

```
aws redshift-data describe-table
    --db-user myuser
    --cluster-identifier mycluster-test
    --database dev
    --schema information_schema
    --table sql_features
```

The following is an example of the response.

```
{
    "ColumnList": [
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": false,
            "length": 2147483647,
            "name": "feature_id",
            "nullable": 1,
            "precision": 2147483647,
            "scale": 0,
            "schemaName": "information_schema",
            "tableName": "sql_features",
```

```
            "typeName": "character_data"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": false,
            "length": 2147483647,
            "name": "feature_name",
            "nullable": 1,
            "precision": 2147483647,
            "scale": 0,
            "schemaName": "information_schema",
            "tableName": "sql_features",
            "typeName": "character_data"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": false,
            "length": 2147483647,
            "name": "sub_feature_id",
            "nullable": 1,
            "precision": 2147483647,
            "scale": 0,
            "schemaName": "information_schema",
            "tableName": "sql_features",
            "typeName": "character_data"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": false,
            "length": 2147483647,
            "name": "sub_feature_name",
            "nullable": 1,
            "precision": 2147483647,
            "scale": 0,
            "schemaName": "information_schema",
            "tableName": "sql_features",
            "typeName": "character_data"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
```

```
            "isSigned": false,
            "length": 2147483647,
            "name": "is_supported",
            "nullable": 1,
            "precision": 2147483647,
            "scale": 0,
            "schemaName": "information_schema",
            "tableName": "sql_features",
            "typeName": "character_data"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": false,
            "length": 2147483647,
            "name": "is_verified_by",
            "nullable": 1,
            "precision": 2147483647,
            "scale": 0,
            "schemaName": "information_schema",
            "tableName": "sql_features",
            "typeName": "character_data"
        },
        {
            "isCaseSensitive": false,
            "isCurrency": false,
            "isSigned": false,
            "length": 2147483647,
            "name": "comments",
            "nullable": 1,
            "precision": 2147483647,
            "scale": 0,
            "schemaName": "information_schema",
            "tableName": "sql_features",
            "typeName": "character_data"
        }
    ]
}
```


List databases in a cluster

To list the databases in a cluster, use the `aws redshift-data list-databases` AWS CLI command.

The following AWS CLI command runs a SQL statement against a cluster to list databases. This example uses the AWS Secrets Manager authentication method.

```
aws redshift-data list-databases

    --secret-arn arn:aws:secretsmanager:us-west-2:123456789012:secret:myuser-secret-
hKgPWn
    --cluster-identifier mycluster-test
    --database dev
```

The following is an example of the response.

```
{
    "Databases": [
        "dev"
    ]
}
```

The following AWS CLI command runs a SQL statement against a cluster to list databases. This example uses the temporary credentials authentication method.

```
aws redshift-data list-databases
    --db-user myuser
    --cluster-identifier mycluster-test
    --database dev
```

The following is an example of the response.

```
{
    "Databases": [
        "dev"
    ]
}
```

List schemas in a database

To list the schemas in a database, use the `aws redshift-data list-schemas` AWS CLI command.

The following AWS CLI command runs a SQL statement against a cluster to list schemas in a database. This example uses the AWS Secrets Manager authentication method.

```
aws redshift-data list-schemas
    --secret-arn arn:aws:secretsmanager:us-west-2:123456789012:secret:myuser-secret-
hKgPWn
    --cluster-identifier mycluster-test
    --database dev
```

The following is an example of the response.

```
{
    "Schemas": [
        "information_schema",
        "pg_catalog",
        "pg_internal",
        "public"
    ]
}
```

The following AWS CLI command runs a SQL statement against a cluster to list schemas in a database. This example uses the temporary credentials authentication method.

```
aws redshift-data list-schemas
    --db-user mysuser
    --cluster-identifier mycluster-test
    --database dev
```

The following is an example of the response.

```
{
    "Schemas": [
        "information_schema",
        "pg_catalog",
        "pg_internal",
        "public"
    ]
}
```

List tables in a database

To list the tables in a database, use the `aws redshift-data list-tables` AWS CLI command.

The following AWS CLI command runs a SQL statement against a cluster to list tables in a database. This example uses the AWS Secrets Manager authentication method.

```
aws redshift-data list-tables
    --secret-arn arn:aws:secretsmanager:us-west-2:123456789012:secret:myuser-secret-
hKgPWn
    --cluster-identifier mycluster-test
    --database dev
    --schema information_schema
```

The following is an example of the response.

```
{
    "Tables": [
        {
            "name": "sql_features",
            "schema": "information_schema",
            "type": "SYSTEM TABLE"
        },
        {
            "name": "sql_implementation_info",
            "schema": "information_schema",
            "type": "SYSTEM TABLE"
        }
}
```

The following AWS CLI command runs a SQL statement against a cluster to list tables in a database. This example uses the temporary credentials authentication method.

```
aws redshift-data list-tables

     --db-user myuser
     --cluster-identifier mycluster-test
     --database dev
```

```
    --schema information_schema
```

The following is an example of the response.

```
{
    "Tables": [
        {
            "name": "sql_features",
            "schema": "information_schema",
            "type": "SYSTEM TABLE"
        },
        {
            "name": "sql_implementation_info",
            "schema": "information_schema",
            "type": "SYSTEM TABLE"
        }
    ]
}
```

Troubleshooting issues for Amazon Redshift Data API

Use the following sections, titled with common error messages, to help troubleshoot problems that you have with the Data API.

Topics

- Packet for query is too large
- Database response exceeded size limit

Packet for query is too large

If you see an error indicating that the packet for a query is too large, generally the result set returned for a row is too large. The Data API size limit is 64 KB per row in the result set returned by the database.

To solve this issue, make sure that each row in a result set is 64 KB or less.

Database response exceeded size limit

If you see an error indicating that the database response has exceeded the size limit, generally the size of the result set returned by the database was too large. The Data API limit is 500 MB in the result set returned by the database.

To solve this issue, make sure that calls to the Data API return 500 MB of data or less. If you need to return more than 500 MB, you can run multiple statement calls with the `LIMIT` clause in your query.

Scheduling Amazon Redshift Data API operations with Amazon EventBridge

You can create rules that match selected events and route them to targets to take action. You can also use rules to take action on a predetermined schedule. For more information, see the Amazon EventBridge User Guide.

To schedule Data API operations with EventBridge, the associated IAM role must trust the principal for CloudWatch Events (events.amazonaws.com). This role should have the equivalent of the managed policy `AmazonEventBridgeFullAccess` attached. It should also have `AmazonRedshiftDataFullAccess` policy permissions that are managed by the Data API. You can create an IAM role with these permissions on the IAM console. When creating a role on the IAM console, choose the AWS service trusted entity for CloudWatch Events. Specify the IAM role in the `RoleArn` JSON value in the EventBridge target. For more information about creating an IAM role, see Creating a Role for an AWS Service (Console) in the *IAM User Guide*.

The `name` of the rule that you create in Amazon EventBridge must match the `StatementName` in the `RedshiftDataParameters`.

The following examples show variations of creating EventBridge rules with a single or multiple SQL statements and with an Amazon Redshift cluster or an Amazon Redshift Serverless workgroup as the data warehouse.

Calling with a single SQL statement and cluster

The following example uses the AWS CLI to create an EventBridge rule that is used to run a SQL statement against an Amazon Redshift cluster.

```
aws events put-rule
```

```
--name test-redshift-cluster-data
--schedule-expression "rate(1 minute)"
```

Then an EventBridge target is created to run on the schedule specified in the rule.

```
aws events put-targets
--cli-input-json file://data.json
```

The input data.json file is as follows. The `Sql` JSON key indicates there is a single SQL statement. The `Arn` JSON value contains a cluster identifier. The `RoleArn` JSON value contains the IAM role used to run the SQL as described previously.

```
{
    "Rule": "test-redshift-cluster-data",
    "EventBusName": "default",
    "Targets": [
        {
            "Id": "2",
            "Arn": "arn:aws:redshift:us-east-1:123456789012:cluster:mycluster",
            "RoleArn": "arn:aws:iam::123456789012:role/Administrator",
            "RedshiftDataParameters": {
                "Database": "dev",
                "DbUser": "root",
                "Sql": "select 1;",
                "StatementName": "test-redshift-cluster-data",
                "WithEvent": true
            }
        }
    ]
}
```

Calling with a single SQL statement and workgroup

The following example uses the AWS CLI to create an EventBridge rule that is used to run a SQL statement against an Amazon Redshift Serverless workgroup.

```
aws events put-rule
--name  test-redshift-serverless-workgroup-data
--schedule-expression "rate(1 minute)"
```

Then an EventBridge target is created to run on the schedule specified in the rule.

```
aws events put-targets
--cli-input-json file://data.json
```

The input data.json file is as follows. The `Sql` JSON key indicates there is a single SQL statement. The `Arn` JSON value contains a workgroup name. The `RoleArn` JSON value contains the IAM role used to run the SQL as described previously.

```
{
    "Rule": "test-redshift-serverless-workgroup-data",
    "EventBusName": "default",
    "Targets": [
        {
            "Id": "2",
            "Arn": "arn:aws:redshift-serverless:us-east-1:123456789012:workgroup/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "RoleArn": "arn:aws:iam::123456789012:role/Administrator",
            "RedshiftDataParameters": {
                "Database": "dev",
                "Sql": "select 1;",
                "StatementName": "test-redshift-serverless-workgroup-data",
                "WithEvent": true
            }
        }
    ]
}
```

Calling with multiple SQL statements and cluster

The following example uses the AWS CLI to create an EventBridge rule that is used to run multiple SQL statements against an Amazon Redshift cluster.

```
aws events put-rule
--name  test-redshift-cluster-data
--schedule-expression "rate(1 minute)"
```

Then an EventBridge target is created to run on the schedule specified in the rule.

```
aws events put-targets
--cli-input-json file://data.json
```

The input data.json file is as follows. The `Sqls` JSON key indicates there are multiple SQL statements. The `Arn` JSON value contains a cluster identifier. The `RoleArn` JSON value contains the IAM role used to run the SQL as described previously.

```
{
    "Rule": "test-redshift-cluster-data",
    "EventBusName": "default",
    "Targets": [
        {
            "Id": "2",
            "Arn": "arn:aws:redshift:us-east-1:123456789012:cluster:mycluster",
            "RoleArn": "arn:aws:iam::123456789012:role/Administrator",
            "RedshiftDataParameters": {
                "Database": "dev",
                "Sqls": ["select 1;", "select 2;", "select 3;"],
                "StatementName": "test-redshift-cluster-data",
                "WithEvent": true
            }
        }
    ]
}
```

Calling with multiple SQL statements and workgroup

The following example uses the AWS CLI to create an EventBridge rule that is used to run multiple SQL statements against an Amazon Redshift Serverless workgroup.

```
aws events put-rule
--name  test-redshift-serverless-workgroup-data
--schedule-expression "rate(1 minute)"
```

Then an EventBridge target is created to run on the schedule specified in the rule.

```
aws events put-targets
--cli-input-json file://data.json
```

The input data.json file is as follows. The `Sqls` JSON key indicates there are multiple SQL statements. The `Arn` JSON value contains a workgroup name. The `RoleArn` JSON value contains the IAM role used to run the SQL as described previously.

```
{
```

```
    "Rule": "test-redshift-serverless-workgroup-data",
    "EventBusName": "default",
    "Targets": [
        {
            "Id": "2",
            "Arn": "arn:aws:redshift-serverless:us-east-1:123456789012:workgroup/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "RoleArn": "arn:aws:iam::123456789012:role/Administrator",
            "RedshiftDataParameters": {
                "Database": "dev",
                "Sqls": ["select 1;", "select 2;", "select 3;"],
                "StatementName": "test-redshift-serverless-workgroup-data",
                "WithEvent": true
            }
        }
    ]
}
```

Monitoring the Data API

Monitoring is an important part of maintaining the reliability, availability, and performance of the Data API and your other AWS solutions. AWS provides the following monitoring tools to watch the Data API, report when something is wrong, and take automatic actions when appropriate:

- Amazon EventBridge can be used to automate your AWS services and respond automatically to system events, such as application availability issues or resource changes. Events from AWS services are delivered to EventBridge in near-real time. You can write simple rules to indicate which events are of interest to you and which automated actions to take when an event matches a rule. For more information, see the Amazon EventBridge User Guide.
- AWS CloudTrail captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. To learn more about how Amazon Redshift is integrated with AWS CloudTrail, see Logging with CloudTrail. For more information about CloudTrail, see the AWS CloudTrail User Guide.

Topics

- Monitoring events for the Amazon Redshift Data API in Amazon EventBridge

Monitoring events for the Amazon Redshift Data API in Amazon EventBridge

You can monitor Data API events in EventBridge, which delivers a stream of real-time data from your own applications, software-as-a-service (SaaS) applications, and AWS services. EventBridge routes that data to targets such as AWS Lambda and Amazon SNS. These events are the same as those that appear in CloudWatch Events, which delivers a near-real time stream of system events that describe changes in AWS resources. Events are sent to the account that contains the Amazon Redshift database. For example, if you assume a role in another account, events are sent to that account. For more information, see Amazon EventBridge events in the *Amazon EventBridge User Guide.*.

Data API events are sent when the `ExecuteStatement` or `BatchExecuteStatement` API operation sets the `WithEvent` option to `true`. The `state` field of the event contains one of the following values:

- ABORTED – The query run was stopped by the user.
- FAILED – The query run failed.
- FINISHED – The query has finished running.

Events are delivered on a guaranteed basis. For more information, see Events from AWS services in the *Amazon EventBridge User Guide*.

Example for Data API finished event

The following example shows an event for the Data API when the `ExecuteStatement` API operation finished. In this example, a statement named `test.testtable` finished running.

```
{
    "version": "0",
    "id": "18e7079c-dd4b-dd64-caf9-e2a31640dab0",
    "detail-type": "Redshift Data Statement Status Change",
    "source": "aws.redshift-data",
    "account": "123456789012",
    "time": "2020-10-01T21:14:26Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:redshift:us-east-1:123456789012:cluster:redshift-cluster-1"
    ],
    "detail": {
        "principal": "arn:aws:iam::123456789012:user/myuser",
```

```
        "statementName": "test.testtable",
        "statementId": "dd2e1ec9-2ee3-49a0-819f-905fa7d75a4a",
        "redshiftQueryId": -1,
        "state": "FINISHED",
        "rows": 1,
        "expireAt": 1601673265
    }
}
```

Using AWS KMS with the Amazon Redshift Data API

When you encrypt your Amazon Redshift cluster or Redshift Serverless workgroup with a customer managed key, the Amazon Redshift Data API uses that same customer managed key to store and encrypt your queries and results.

The Data API encrypts your data by default to protect sensitive information, such as query text and query results. It uses AWS KMS encryption keys owned by AWS for this protection.

Default encryption for data at rest reduces operational overhead and complexity when you protect sensitive data. This approach helps you build secure applications that meet strict encryption compliance and regulatory requirements.

Using grants in AWS KMS

The Data API requires a grant to use your customer managed key.

When you call `ExecuteStatement` or `BatchExecuteStatement` against a cluster encrypted with a customer managed key, Amazon Redshift creates a grant on your behalf by sending a `CreateGrant` request to AWS KMS. AWS KMS uses grants to give the Data API access to a KMS key in your account.

The Data API requires the grant to use your customer managed key for the following operations:

- Send `Encrypt` requests to AWS KMS to encrypt query metadata with your customer managed key.
- Send `GenerateDataKey` requests to AWS KMS to generate data keys encrypted by your customer managed key.
- Send `Decrypt` requests to AWS KMS to decrypt the encrypted data keys so they can encrypt your data.

You can revoke access to the grant or remove Amazon Redshift access to your customer managed key at any time. If you do, the Data API can no longer access data encrypted by your customer managed key, which affects operations that depend on that data. For example, if you try to retrieve query results or track query status after revoking the grant, the Data API returns an `AccessDeniedException`.

Key policies for your customer managed key

Key policies control access to your customer managed key. Every customer managed key must have exactly one key policy, which contains statements that determine who can use the key and how they can use it. When you create your customer managed key, you can specify a key policy. For more information, see Customer managed keys in the *AWS Key Management Service Developer Guide*.

To use your customer managed keys with the Data API, you must first allow access to Amazon Redshift. The following API operations must be permitted in the key policy:

- `kms:CreateGrant` – Adds a grant to a customer managed key. Grants control access to a specified AWS KMS key, which allows access to grant operations that Amazon Redshift requires. For more information, see Using grants in AWS KMS.

The following is an example key policy:

```
"Statement":[
   {
      "Sid":"Allow access to principals authorized to use Amazon Redshift",
      "Effect":"Allow",
      "Principal":{
         "AWS":"*"
      },
      "Action":[
         "kms:DescribeKey",
         "kms:CreateGrant"
      ],
      "Resource":"*",
      "Condition":{
         "StringEquals":{
            "kms:ViaService":"redshift.amazonaws.com",
            "kms:CallerAccount":"111122223333"
         }
      }
```

```
   },
   {
      "Sid":"AllowKeyAdministratorsAccess",
      "Effect":"Allow",
      "Principal":{
         "AWS":"arn:aws:iam::111122223333:role/ExampleAdminRole"
      },
      "Action":"kms:*",
      "Resource":"*"
   },
   {
      "Sid":"AllowKeyUseForExampleRole",
      "Effect":"Allow",
      "Principal":{
         "AWS":"arn:aws:iam::111122223333:role/ExampleUserRole"
      },
      "Action":[
         "kms:Encrypt",
         "kms:Decrypt",
         "kms:ReEncrypt*",
         "kms:GenerateDataKey*",
         "kms:DescribeKey"
      ],
      "Resource":"*"
   }
]
```

Data API encryption context

An encryption context is an optional set of key-value pairs that contains additional contextual information about the data. AWS KMS uses the encryption context as additional authenticated data to support authenticated encryption. When you include an encryption context in a request to encrypt data, AWS KMS binds the encryption context to the encrypted data. To decrypt the data, you must include the same encryption context in the request.

The Data API uses the same three encryption context key-value pairs in all AWS KMS cryptographic operations for provisioned clusters:

- `aws:redshift:arn` – The cluster's Amazon Resource Name (ARN)
- `aws:redshift:createtime` – The timestamp when you requested cluster creation
- `serviceName` – `RedshiftDataAPI`

```
"EncryptionContextSubset": {
    "aws:redshift:arn": "arn:aws:redshift:us-east-1:123456789012:cluster:redshift-
cluster",
    "aws:redshift:createtime": "20250815T0000Z",
    "serviceName": "RedshiftDataAPI",
}
```

The Data API uses two encryption context key-value pairs in all AWS KMS cryptographic operations for serverless workgroups:

- `aws:redshift-serverless:arn` – The namespace's Amazon Resource Name (ARN)
- `serviceName` – RedshiftDataAPI

```
"EncryptionContextSubset": {
    "aws:redshift-serverless:arn": "arn:aws:redshift-serverless:us-
east-1:123456789012:namespace:12345678-1234-1234-1234-123456789012",
    "serviceName": "RedshiftDataAPI"
}
```

For more information about encryption, see Introduction to the cryptographic details of AWS KMS. For more information about the Amazon Redshift and AWS KMS integration, see How Amazon Redshift uses AWS KMS.

Using Amazon Sagemaker Unified Studio to query your databases in Amazon Redshift and the SageMaker lakehouse

Amazon SageMaker Unified Studio provides an off-console development environment and supports SQL analytics on data in the SageMaker lakehouse, Amazon Redshift, and Amazon Athena for SQL analytics. Navigate to Amazon SageMaker Unified Studio using the URL from your admin and use your SSO or AWS credentials to log in. For more information about setting up your first project, see Getting started in the *Amazon SageMaker Unified Studio User Guide*.

In Amazon SageMaker Unified Studio, you can perform SQL analytics by running Amazon Redshift and Amazon Athena with the query editor. Use the query editor to write and run queries, view results, and share your work with your team. Run queries against your Redshift data warehouses in your AWS accounts (within the same account and across your other AWS accounts), build SQL queries for both Redshift and Athena using the same interface and schedule the SQL queries using

Amazon Managed Workflows for Apache Airflow. You can also use Amazon Q generative SQL to generate SQL from natural language.

Amazon Redshift parameter groups

In Amazon Redshift, you associate a parameter group with each cluster that you create. A *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. When you launch a cluster, you must associate it with a parameter group. If you want to change the parameter group later, you can modify the cluster and choose a different parameter group.

Each parameter group has several parameters to configure settings for the database. The list of available parameters depends on the parameter group family to which the parameter group belongs. The default parameter group family is `redshift-2.0`.

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. The format of the default parameter group name is `default.`*`parameter_group_family`*. For example, the default parameter group for the `redshift-2.0` parameter group family is `default.redshift-2.0`.

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.

> **Note**
>
> The Amazon Redshift console does not display the `source` of each parameter. You must use the Amazon Redshift API, the AWS CLI, or one of the AWS SDKs to view the `source`.

For parameter groups that you create, you can modify a parameter value at any time, or you can reset all parameter values to their defaults. You can also associate a different parameter group with a cluster. In some cases, you might modify parameter values in a parameter group that is already associated with a cluster or associate a different parameter group with a cluster. In these cases, you might need to restart the cluster for the updated parameter values to take effect. If the cluster fails and is restarted by Amazon Redshift, your changes are applied at that time. Changes aren't applied

if your cluster is restarted during maintenance. For more information, see WLM dynamic and static properties.

Default parameter values

> **Note**
>
> As of January 10, 2025, the default value for the `require_ssl` parameter is true. If you don't want your cluster to require SSL, you can use a custom parameter group when creating the cluster, or modify the cluster to associate it with a custom parameter group after creating the cluster with the default.

The following table shows the default parameter values at a glance with links to more in-depth information about each parameter. These are the default values for the `redshift-2.0` parameter group family.

Parameter name	Value	More information
auto_analyze	true	auto_analyze in the *Amazon Redshift Database Developer Guide*
auto_mv	true	Automated materialized views in the Amazon Redshift Database Developer Guide
datestyle	ISO, MDY	datestyle in the *Amazon Redshift Database Developer Guide*
enable_case_sensitive_identifier	false	enable_case_sensitive_identifier in the *Amazon Redshift Database Developer Guide*
enable_large_strings_opt_in	"" or not set	enable_large_strings_opt_in in the *Amazon Redshift Database Developer Guide*
enable_user_activity_logging	false	Database audit logging in this guide

Parameter name	Value	More information
extra_float_digits	0	extra_float_digits in the *Amazon Redshift Database Developer Guide*
max_concurrency_scaling_clusters	1	max_concurrency_scaling_clusters in the *Amazon Redshift Database Developer Guide*
query_group	default	query_group in the *Amazon Redshift Database Developer Guide*
require_ssl	true	Configuring security options for connections in this guide
search_path	$user, public	search_path in the *Amazon Redshift Database Developer Guide*
statement_timeout	0	statement_timeout in the *Amazon Redshift Database Developer Guide*
wlm_json_configuration	[{"auto_wlm":true}]	Workload management in this guide
use_fips_ssl	false	Enable FIPS-compliant SSL mode only if your system is required to be FIPS-compliant.

Note

The `max_cursor_result_set_size` parameter is deprecated. For more information about cursor result set size, see Cursor constraints in the *Amazon Redshift Database Developer Guide*.

You can temporarily override a parameter by using the SET command in the database. The SET command overrides the parameter for the duration of your current session only. In addition to the parameters listed in the preceding table, you can also temporarily adjust the slot count by setting `wlm_query_slot_count` in the database. The `wlm_query_slot_count` parameter is not available for configuration in parameter groups. For more information about adjusting

the slot count, see wlm_query_slot_count in the *Amazon Redshift Database Developer Guide*. For more information about temporarily overriding the other parameters, see Modifying the server configuration in the *Amazon Redshift Database Developer Guide*.

Workload management

In Amazon Redshift, you use workload management (WLM) to define the number of query queues that are available, and how queries are routed to those queues for processing. WLM is part of parameter group configuration. A cluster uses the WLM configuration that is specified in its associated parameter group.

When you create a parameter group, the default WLM configuration contains one queue that can run up to five queries concurrently. You can add additional queues and configure WLM properties in each of them if you want more control over query processing. Each queue that you add has the same default WLM configuration until you configure its properties.

When you add additional queues, the last queue in the configuration is the *default queue*. Unless a query is routed to another queue based on criteria in the WLM configuration, it is processed by the default queue. You can specify mode and concurrency level (query slots) for the default queue, but you can't specify user groups or query groups for the default queue.

As with other parameters, you cannot modify the WLM configuration in the default parameter group. Clusters associated with the default parameter group always use the default WLM configuration. To modify the WLM configuration, create a new parameter group and then associate that parameter group with any clusters that require your custom WLM configuration.

WLM dynamic and static properties

The WLM configuration properties are either dynamic or static. You can apply dynamic properties to the database without a cluster reboot, but static properties require a cluster reboot for changes to take effect. For more information about static and dynamic properties, see WLM dynamic and static configuration properties.

Properties for the WLM configuration parameter

You can configure WLM by using the Amazon Redshift console, the AWS CLI, the Amazon Redshift API, or one of the AWS SDKs. WLM configuration uses several properties to define queue behavior, such as memory allocation across queues, the number of queries that can run concurrently in a queue, and so on.

Note

The following properties appear with their Amazon Redshift console names, with the corresponding JSON property names in the descriptions.

The following table summarizes whether a property is applicable to automatic WLM or manual WLM.

WLM property	Automatic WLM	Manual WLM
Auto WLM	Yes	Yes
Enable short query acceleration	Yes	Yes
Maximum run time for short queries	Yes	Yes
Priority	Yes	No
Queue type	Yes	Yes
Queue name	Yes	Yes
Concurrency Scaling mode	Yes	Yes
Concurrency	No	Yes
User groups	Yes	Yes
User group wildcard	Yes	Yes
Query groups	Yes	Yes
Query group wildcard	Yes	Yes
User roles	Yes	Yes
User role wildcard	Yes	Yes

WLM property	Automatic WLM	Manual WLM
Timeout	No	Deprecated
Memory	No	Yes
Query Monitoring Rules	Yes	Yes

The following list describes the WLM properties that you can configure.

Auto WLM

Auto WLM set to `true` enables automatic WLM. Automatic WLM sets the values for **Concurrency on main** and **Memory (%)** to `Auto`. Amazon Redshift manages query concurrency and memory allocation. The default is `true`.

JSON property: `auto_wlm`

Enable short query acceleration

Short query acceleration (SQA) prioritizes selected short-running queries ahead of longer-running queries. SQA executes short-running queries in a dedicated space, so that SQA queries aren't forced to wait in queues behind longer queries. With SQA, short-running queries begin executing more quickly and users see results sooner. When you enable SQA, you can also specify the maximum run time for short queries. To enable SQA, specify `true`. The default is `false`. This setting is applied for each parameter group rather than queue.

JSON property: `short_query_queue`

Maximum run time for short queries

When you enable SQA, you can specify 0 to let WLM dynamically set the maximum run time for short queries. Alternatively, you can specify a value of 1–20 seconds, in milliseconds. The default value is `0`.

JSON property: `max_execution_time`

Priority

Priority sets the priority of queries that run in a queue. To set the priority, **WLM mode** must be set to **Auto WLM**; that is, `auto_wlm` must be `true`. Priority values can be `highest`, `high`, `normal`, `low`, and `lowest`. The default is `normal`.

JSON property: `priority`

Queue type

Queue type designates a queue as used either by **Auto WLM** or **Manual WLM**. Set `queue_type` to either `auto` or `manual`. If not specified, the default is `manual`.

JSON property: `queue_type`

Queue name

The name of the queue. You can set the name of the queue based on your business needs. Queue names must be unique within an WLM configuration, are up to 64 alphanumeric characters, underscores or spaces, and can't contain quotation marks. For example, if you have a queue for your ETL queries, you might name it `ETL queue`. This name is used in metrics, system table values, and the Amazon Redshift console to identify the queue. Queries and reports that use the name from these sources need to be able to handle changes of the name. Previously, the queue names were generated by Amazon Redshift. The default names of queues are `Queue 1`, `Queue 2`, to the last queue named `Default queue`.

> **Important**
>
> If you change a queue name, the `QueueName` dimension value of WLM queue metrics (such as, WLMQueueLength, WLMQueueWaitTime, WLMQueriesCompletedPerSecond, WLMQueryDuration, WLMRunningQueries, and so on) also changes. So, if you change the name of a queue, you might need to change CloudWatch alarms you have set up.

JSON property: `name`

Concurrency Scaling mode

To enable concurrency scaling on a queue, set **Concurrency Scaling mode** to `auto`. When the number of queries routed to a queue exceeds the queue's configured concurrency, eligible queries go to the scaling cluster. When slots become available, queries run on the main cluster. The default is `off`.

JSON property: `concurrency_scaling`

Concurrency

The number of queries that can run concurrently in a manual WLM queue. This property only applies to manual WLM. If concurrency scaling is enabled, eligible queries go to a scaling cluster

when a queue reaches the concurrency level (query slots). If concurrency scaling isn't enabled, queries wait in the queue until a slot becomes available. The range is between 1 and 50.

JSON property: `query_concurrency`

User Groups

A comma-separated list of user group names. When members of the user group run queries in the database, their queries are routed to the queue that is associated with their user group.

JSON property: `user_group`

User Group Wildcard

A Boolean value that indicates whether to enable wildcards for user groups. If this is 0, wildcards are disabled; if this is 1, wildcards are enabled. When wildcards are enabled, you can use "*" or "?" to specify multiple user groups when running queries. For more information, see Wildcards.

JSON property: `user_group_wild_card`

Query Groups

A comma-separated list of query groups. When members of the query group run queries in the database, their queries are routed to the queue that is associated with their query group.

JSON property: `query_group`

Query Group Wildcard

A Boolean value that indicates whether to enable wildcards for query groups. If this is 0, wildcards are disabled; if this is 1, wildcards are enabled. When wildcards are enabled, you can use "*" or "?" to specify multiple query groups when running queries. For more information, see Wildcards.

JSON property: `query_group_wild_card`

User Roles

A comma-separated list of user roles. When members with that user role run queries in the database, their queries are routed to the queue that is associated with their user role. For more information about user roles, see Role-based access control (RBAC).

JSON property: `user_role`

User Roles Wildcard

A Boolean value that indicates whether to enable wildcards for query groups. If this is 0, wildcards are disabled; if this is 1, wildcards are enabled. When wildcards are enabled, you can use "*" or "?" to specify multiple query groups when running queries. For more information, see Wildcards.

JSON property: `user_role_wild_card`

Timeout (ms)

WLM timeout (`max_execution_time`) is deprecated. It is not available when using automatic WLM. Instead, create a query monitoring rule (QMR) using `query_execution_time` to limit the elapsed execution time for a query. For more information, see WLM query monitoring rules.

The maximum time, in milliseconds, that queries can run before being canceled. In some cases, a read-only query, such as a SELECT statement, might be canceled due to a WLM timeout. In these cases, WLM attempts to route the query to the next matching queue based on the WLM queue assignment rules. If the query doesn't match any other queue definition, the query is canceled; it isn't assigned to the default queue. For more information, see WLM query queue hopping. WLM timeout doesn't apply to a query that has reached the `returning` state. To view the state of a query, see the STV_WLM_QUERY_STATE system table.

JSON property: `max_execution_time`

Memory (%)

The percentage of memory to allocate to the queue. If you specify a memory percentage for at least one of the queues, you must specify a percentage for all other queues, up to a total of 100 percent. If your memory allocation is below 100 percent across all of the queues, the unallocated memory is managed by the service. The service can temporarily give this unallocated memory to a queue that requests additional memory for processing.

JSON property: `memory_percent_to_use`

Query Monitoring Rules

You can use WLM query monitoring rules to continuously monitor your WLM queues for queries based on criteria, or predicates, that you specify. For example, you might monitor queries that tend to consume excessive system resources, and then initiate a specified action when a query exceeds your specified performance boundaries.

> **Note**
>
> If you choose to create rules programmatically, we strongly recommend using the console to generate the JSON that you include in the parameter group definition.

You associate a query monitoring rule with a specific query queue. You can have up to 25 rules per queue, and the total limit for all queues is 25 rules.

JSON property: `rules`

JSON properties hierarchy:

```
rules
    rule_name
    predicate
        metric_name
        operator
        value
    action
        value
```

For each rule, you specify the following properties:

- `rule_name` – Rule names must be unique within WLM configuration. Rule names can be up to 32 alphanumeric characters or underscores, and can't contain spaces or quotation marks.
  - `predicate` – You can have up to three predicates per rule. For each predicate, specify the following properties.
    - `metric_name` – For a list of metrics, see Query monitoring metrics in the *Amazon Redshift Database Developer Guide*.
    - `operator` – Operations are =, <, and >.
    - `value` – The threshold value for the specified metric that triggers an action.
- `action` – Each rule is associated with one action. Valid actions are:
  - `log`
  - hop (only available with manual WLM)
  - `abort`
  - `change_query_priority` (only available with automatic WLM)

The following example shows the JSON for a WLM query monitoring rule named `rule_1`, with two predicates and the action hop.

```
"rules": [
          {
            "rule_name": "rule_1",
            "predicate": [
              {
                "metric_name": "query_execution_time",
                "operator": ">",
                "value": 100000
              },
              {
                "metric_name": "query_blocks_read",
                "operator": ">",
                "value": 1000
              }
            ],
            "action": "hop"
          }
        ]
```

For more information about each of these properties and strategies for configuring query queues, see Implementing workload management in the *Amazon Redshift Database Developer Guide*.

Configuring the WLM parameter using the AWS CLI

To configure WLM, you modify the `wlm_json_configuration` parameter. The maximum size of the `wlm_json_configuration` property value is 8000 characters. The value is formatted in JavaScript Object Notation (JSON). If you configure WLM by using the AWS CLI, Amazon Redshift API, or one of the AWS SDKs, use the rest of this section to learn how to construct the JSON structure for the `wlm_json_configuration` parameter.

Note

If you configure WLM by using the Amazon Redshift console, you don't need to understand JSON formatting because the console provides an easy way to add queues and configure their properties. For more information about configuring WLM by using the console, see Modifying a parameter group.

Example

The following example is the default WLM configuration, which defines one queue with automatic WLM.

```
{
   "auto_wlm": true
}
```

Example

The following example is a custom WLM configuration, which defines one manual WLM queue with a concurrency level (query slots) of five.

```
{
   "query_concurrency":5
}
```

Syntax

The default WLM configuration is very simple, with only queue and one property. You can add more queues and configure multiple properties for each queue in the JSON structure. The following syntax represents the JSON structure that you use to configure multiple queues with multiple properties:

```
[
   {
      "ParameterName":"wlm_json_configuration", "ParameterValue":
         "[
             {
                "q1_first_property_name":"q1_first_property_value",
                "q1_second_property_name":"q1_second_property_value",
                ...
             },
             {
                "q2_first_property_name":"q2_first_property_value",
                "q2_second_property_name":"q2_second_property_value",
                ...
             }
             ...

         ]"
```

```
    }
]
```

In the preceding example, the representative properties that begin with **q1** are objects in an array for the first queue. Each of these objects is a name/value pair; `name` and `value` together set the WLM properties for the first queue. The representative properties that begin with **q2** are objects in an array for the second queue. If you require more queues, you add another array for each additional queue and set the properties for each object.

When you modify the WLM configuration, you must include in the entire structure for your queues, even if you only want to change one property within a queue. This is because the entire JSON structure is passed in as a string as the value for the `wlm_json_configuration` parameter.

Formatting the AWS CLI command

The `wlm_json_configuration` parameter requires a specific format when you use the AWS CLI. The format that you use depends on your client operating system. Operating systems have different ways to enclose the JSON structure so it's passed correctly from the command line. For details on how to construct the appropriate command in the Linux, macOS, and Windows operating systems, see the sections following. For more information about the differences in enclosing JSON data structures in the AWS CLI in general, see Quoting strings in the *AWS Command Line Interface User Guide*.

Examples

The following example command configures manual WLM for a parameter group called `example-parameter-group`. The configuration enables short-query acceleration with a maximum run time for short queries set to 0, which instructs WLM to set the value dynamically. The `ApplyType` setting is `dynamic`. This setting means that any changes made to dynamic properties in the parameter are applied immediately unless other static changes have been made to the configuration. The configuration defines three queues with the following:

- The first queue enables users to specify `report` as a label (as specified in the `query_group` property) in their queries to help in routing queries to that queue. Wildcard searches are enabled for the `report*` label, so the label doesn't need to be exact for queries to be routed to the queue. For example, `reports` and `reporting` both match this query group. The queue is allocated 25 percent of the total memory across all queues, and can run up to four queries at the same time. Queries are limited to a maximum time of 20000 milliseconds (ms). mode is set to auto, so when the queue's query slots are full eligible queries are sent to a scaling cluster.

- The second queue enables users who are members of admin or dba groups in the database to have their queries routed to the queue for processing. Wildcard searches are disabled for user groups, so users must be matched exactly to groups in the database in order for their queries to be routed to the queue. The queue is allocated 40 percent of the total memory across all queues, and it can run up to five queries at the same time. mode is set to off, so all queries sent by members of the admin or dba groups run on the main cluster.
- The last queue in the configuration is the default queue. This queue is allocated 35 percent of the total memory across all queues, and it can process up to five queries at a time. mode is set to auto.

Note

The example is shown on several lines for demonstration purposes. Actual commands should not have line breaks.

```
aws redshift modify-cluster-parameter-group
--parameter-group-name example-parameter-group
--parameters
'[
  {
    "query_concurrency": 4,
    "max_execution_time": 20000,
    "memory_percent_to_use": 25,
    "query_group": ["report"],
    "query_group_wild_card": 1,
    "user_group": [],
    "user_group_wild_card": 0,
    "user_role": [],
    "user_role_wild_card": 0,
    "concurrency_scaling": "auto",
    "queue_type": "manual"
  },
  {
    "query_concurrency": 5,
    "memory_percent_to_use": 40,
    "query_group": [],
    "query_group_wild_card": 0,
    "user_group": [
      "admin",
```

```
      "dba"
    ],
    "user_group_wild_card": 0,
    "user_role": [],
    "user_role_wild_card": 0,
    "concurrency_scaling": "off",
    "queue_type": "manual"
  },
  {
    "query_concurrency": 5,
    "query_group": [],
    "query_group_wild_card": 0,
    "user_group": [],
    "user_group_wild_card": 0,
    "user_role": [],
    "user_role_wild_card": 0,
    "concurrency_scaling": "auto",
    "queue_type": "manual"
  },
  {"short_query_queue": true}
]'
```

The following is an example of configuring WLM query monitoring rules for an automatic WLM configuration. The example creates a parameter group named `example-monitoring-rules`. The configuration defines the same three queues as the previous example, but the `query_concurrency` and `memory_percent_to_use` are not specified anymore. The configuration also adds the following rules and query priorities:

- The first queue defines a rule named `rule_1`. The rule has two predicates: `query_cpu_time > 10000000` and `query_blocks_read > 1000`. The rule action is `log`. The priority of this queue is `Normal`.
- The second queue defines a rule named `rule_2`. The rule has two predicates: `query_execution_time > 600000000` and `scan_row_count > 1000000000`. The rule action is `abort`. The priority of this queue is `Highest`.
- The last queue in the configuration is the default queue. The priority of this queue is `Low`.

Note

The example is shown on several lines for demonstration purposes. Actual commands should not have line breaks.

```
aws redshift modify-cluster-parameter-group
--parameter-group-name example-monitoring-rules
--parameters
'[ {
  "query_group" : [ "report" ],
  "query_group_wild_card" : 1,
  "user_group" : [ ],
  "user_group_wild_card" : 0,
  "user_role": [ ],
  "user_role_wild_card": 0,
  "concurrency_scaling" : "auto",
  "rules" : [{
    "rule_name": "rule_1",
    "predicate": [{
      "metric_name": "query_cpu_time",
      "operator": ">",
      "value": 1000000 },
      { "metric_name": "query_blocks_read",
      "operator": ">",
      "value": 1000
    } ],
    "action" : "log"
  } ],
   "priority": "normal",
   "queue_type": "auto"
}, {
  "query_group" : [ ],
  "query_group_wild_card" : 0,
  "user_group" : [ "admin", "dba" ],
  "user_group_wild_card" : 0,
  "user_role": [ ],
  "user_role_wild_card": 0,
  "concurrency_scaling" : "off",
  "rules" : [ {
    "rule_name": "rule_2",
    "predicate": [
```

```
      {"metric_name": "query_execution_time",
      "operator": ">",
      "value": 600000000},
      {"metric_name": "scan_row_count",
      "operator": ">",
      "value": 1000000000}],
      "action": "abort"}],
   "priority": "high",
   "queue_type": "auto"

}, {
  "query_group" : [ ],
  "query_group_wild_card" : 0,
  "user_group" : [ ],
  "user_group_wild_card" : 0,
  "user_role": [ ],
  "user_role_wild_card": 0,
  "concurrency_scaling" : "auto",
  "priority": "low",
  "queue_type": "auto",
  "auto_wlm": true
}, {
  "short_query_queue" : true
} ]'
```

Configuring WLM by using the AWS CLI in the command line with a JSON file

You can modify the `wlm_json_configuration` parameter using the AWS CLI and pass in the value of the `parameters` argument as a JSON file.

```
aws redshift modify-cluster-parameter-group --parameter-group-name
 myclusterparaametergroup --parameters file://modify_pg.json
```

The arguments for `--parameters` are stored in file `modify_pg.json`. The file location is specified in the format for your operating system. For more information, see Loading parameters from a file. The following shows examples of the content of the `modify_pg.json` JSON file.

```
[
    {
        "ParameterName": "wlm_json_configuration",
```

```
        "ParameterValue": "[{\"user_group\":\"example_user_group1\",\"query_group\":
 \"example_query_group1\", \"query_concurrency\":7},{\"query_concurrency\":5}]"
    }
]
```

```
[
    {
        "ParameterName": "wlm_json_configuration",
        "ParameterValue": "[{\"query_group\":[\"reports\"],\"query_group_wild_card\":0,
\"query_concurrency\":4,\"max_execution_time\":20000,\"memory_percent_to_use\":25},
{\"user_group\":[\"admin\",\"dba\"],\"user_group_wild_card\":1,\"query_concurrency\":5,
\"memory_percent_to_use\":40},{\"query_concurrency\":5,\"memory_percent_to_use\":35},
{\"short_query_queue\": true, \"max_execution_time\": 5000 }]",
        "ApplyType": "dynamic"
    }
]
```

Rules for configuring WLM by using the AWS CLI in the command line on the Linux and macOS operating systems

Follow these rules to run an AWS CLI command with parameters on one line:

- The entire JSON structure must be enclosed in single quotation marks (') and one set of brackets ([ ]).
- All parameter names and parameter values must be enclosed in double quotation marks (").
- Within the `ParameterValue` value, you must enclose the entire nested structure in double-quotation marks (") and brackets ([ ]).
- Within the nested structure, each of the properties and values for each queue must be enclosed in curly braces ({ }).
- Within the nested structure, you must use the backslash (\) escape character before each double-quotation mark (").
- For name/value pairs, a colon (:) separates each property from its value.
- Each name/value pair is separated from another by a comma (,).
- Multiple queues are separated by a comma (,) between the end of one queue's curly brace (}) and the beginning of the next queue's curly brace ({).

Rules for configuring WLM by using the AWS CLI in Windows PowerShell on Microsoft Windows operating systems

Follow these rules to run an AWS CLI command with parameters on one line:

- The entire JSON structure must be enclosed in single quotation marks (') and one set of brackets ([ ]).
- All parameter names and parameter values must be enclosed in double quotation marks (").
- Within the `ParameterValue` value, you must enclose the entire nested structure in double-quotation marks (") and brackets ([ ]).
- Within the nested structure, each of the properties and values for each queue must be enclosed in curly braces ({ }).
- Within the nested structure, you must use the backslash (\) escape character before each double-quotation mark (") and its backslash (\) escape character. This requirement means that you will use three backslashes and a double quotation mark to make sure that the properties are passed in correctly (\\\").
- For name/value pairs, a colon (:) separates each property from its value.
- Each name/value pair is separated from another by a comma (,).
- Multiple queues are separated by a comma (,) between the end of one queue's curly brace (}) and the beginning of the next queue's curly brace ({).

Rules for configuring WLM by using the command prompt on Windows operating systems

Follow these rules to run an AWS CLI command with parameters on one line:

- The entire JSON structure must be enclosed in double-quotation marks (") and one set of brackets ([ ]).
- All parameter names and parameter values must be enclosed in double quotation marks (").
- Within the `ParameterValue` value, you must enclose the entire nested structure in double-quotation marks (") and brackets ([ ]).
- Within the nested structure, each of the properties and values for each queue must be enclosed in curly braces ({ }).
- Within the nested structure, you must use the backslash (\) escape character before each double-quotation mark (") and its backslash (\) escape character. This requirement means that you will use three backslashes and a double quotation mark to make sure that the properties are passed in correctly (\\\").

- For name/value pairs, a colon (:) separates each property from its value.
- Each name/value pair is separated from another by a comma (,).
- Multiple queues are separated by a comma (,) between the end of one queue's curly brace (}) and the beginning of the next queue's curly brace ({).

Creating a parameter group

If you want to set parameter values that are different from the default parameter group, you can create your own parameter group,

To create a parameter group

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Configurations**, then choose **Workload management** to display the **Workload management** page.
3. Choose **Create** to display the **Create parameter group** window.
4. Enter a value for **Parameter group name** and **Description**.
5. Choose **Create** to create the parameter group.

Modifying a parameter group

You can view any of your parameter groups to see a summary of the values for parameters and workload management (WLM) configuration. You can modify parameters to change the parameter settings and WLM configuration properties.

> **Note**
>
> You can't modify the default parameter group.

AWS Management Console

In the console, group parameters appear on the **Parameters** tab, and **Workload queues** appear on the **Workload Management** tab.

To modify a parameter group

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Configurations**, then choose **Workload management** to display the **Workload management** page.
3. Choose the parameter group that you want to modify to display the details page, with tabs for **Parameters** and **Workload management**.
4. Choose the **Parameters** tab to view the current parameter settings.
5. Choose **Edit parameters** to enable changing settings for these parameters:

   - `auto_analyze`
   - `auto_mv`
   - `datestyle`
   - `enable_case_sensitive_identifier`
   - `enable_user_activity_logging`
   - `extra_float_digits`
   - `max_concurrency_scaling_clusters`
   - `max_cursor_result_set_size`
   - `query_group`
   - `require_ssl`
   - `search_path`
   - `statement_timeout`
   - `use_fips_ssl`

   For more information about these parameters, see Amazon Redshift parameter groups.
6. Enter your changes and then choose **Save** to update the parameter group.

To modify the WLM configuration for a parameter group

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.

2. On the navigation menu, choose **Configurations**, then choose **Workload management** to display the **Workload management** page.
3. Choose the parameter group that you want to modify to display the details page with tabs for **Parameters** and **Workload management**.
4. Choose the **Workload management** tab to view the current WLM configuration.
5. Choose **Edit workload queues** to edit the WLM configuration.
6. (Optional) Select **Enable short query acceleration** to enable short query acceleration (SQA).

   When you enable SQA, **Maximum run time for short queries (1 to 20 seconds)** is set to **Dynamic** by default. To set the maximum runtime to a fixed value, choose a value of 1–20.
7. Do one or more of the following to modify the queue configuration:

   - Choose **Switch WLM mode** to choose between **Automatic WLM** and **Manual WLM**.

     With **Automatic WLM**, the **Memory** and **Concurrency on main** values are set to **auto**.
   - To create a queue, choose **Edit workload queues**, then choose **Add Queue**.
   - To modify a queue, change property values in the table. Depending on the type of queue, properties can include the following:
     - **Queue name** can be changed.
     - **Memory (%)**
     - **Concurrency on main** cluster
     - **Concurrency scaling mode** can be **off** or **auto**
     - **Timeout (ms)**
     - **User groups**
     - **Query groups**
     - **User roles**

     For more information about these properties, see Properties for the WLM configuration parameter.

     Important

     If you change a queue name, the `QueueName` dimension value of WLM queue metrics (such as, WLMQueueLength, WLMQueueWaitTime,

WLMQueriesCompletedPerSecond, WLMQueryDuration, WLMRunningQueries, and so on) also changes. So, if you change the name of a queue, you might need to change CloudWatch alarms you have set up.

- To change the order of queues, choose the **Up** and **Down** arrow buttons.
- To delete a queue, choose **Delete** in the queue's row in the table.

8. (Optional) Select **Defer dynamic changes until reboot** to have the changes applied to clusters after their next reboot.

> **Note**
>
> Some changes require a cluster reboot regardless of this setting. For more information, see WLM dynamic and static properties.

9. Choose **Save**.

AWS CLI

To configure Amazon Redshift parameters by using the AWS CLI, you use the modify-cluster-parameter-group command for a specific parameter group. You specify the parameter group to modify in `parameter-group-name`. You use the `parameters` parameter (for the `modify-cluster-parameter-group` command) to specify name/value pairs for each parameter that you want to modify in the parameter group.

> **Note**
>
> There are special considerations when configuring the `wlm_json_configuration` parameter by using the AWS CLI. The examples in this section apply to all of the parameters except `wlm_json_configuration`. For more information about configuring `wlm_json_configuration` by using the AWS CLI, see Workload management.

After you modify parameter values, you must reboot any clusters that are associated with the modified parameter group. The cluster status displays `applying` for `ParameterApplyStatus` while the values are being applied, and then `pending-reboot` after the values have been

applied. After you reboot, the databases in your cluster begin to use the new parameter values. For more information about rebooting clusters, see Rebooting a cluster.

> **Note**
>
> The `wlm_json_configuration` parameter contains some properties that are dynamic and do not require you to reboot associated clusters for the changes to be applied. For more information about dynamic and static properties, see WLM dynamic and static properties.

The following syntax shows how to use the `modify-cluster-parameter-group` command to configure a parameter. You specify *`parameter_group_name`* and replace both *`parameter_name`* and *`parameter_value`* with an actual parameter to modify and a value for that parameter. If you want to modify more than one parameter at the same time, separate each parameter and value set from the next with a space.

```
aws redshift modify-cluster-parameter-group --parameter-group-
name parameter_group_name --parameters
 ParameterName=parameter_name,ParameterValue=parameter_value
```

The following example shows how to configure the `statement_timeout` and `enable_user_activity_logging` parameters for the `myclusterparametergroup` parameter group.

> **Note**
>
> For readability purposes, the example is displayed on several lines, but in the actual AWS CLI this is one line.

```
aws redshift modify-cluster-parameter-group
--parameter-group-name myclusterparametergroup
--parameters ParameterName=statement_timeout,ParameterValue=20000
 ParameterName=enable_user_activity_logging,ParameterValue=true
```

Creating a query monitoring rule

You can use the Amazon Redshift console to create and modify WLM query monitoring rules. Query monitoring rules are part of the WLM configuration parameter for a parameter group. If you modify a query monitoring rule (QMR), the change happens automatically without the need to modify the cluster. For more information, see WLM query monitoring rules.

When you create a rule, you define the rule name, one or more predicates, and an action.

When you save WLM configuration that includes a rule, you can view the JSON code for the rule definition as part of the JSON for the WLM configuration parameter.

To create a query monitoring rule

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Configurations**, then choose **Workload management** to display the **Workload management** page.
3. Choose the parameter group that you want to modify to display the details page with tabs for **Parameters** and **Workload management**.
4. Choose the **Workload management** tab, and choose **Edit workload queues** to edit the WLM configuration.
5. Add a new rule either by using a predefined template or from scratch.

   To use a predefined template, do the following:

   1. Choose **Add rule from template** in the **Query monitoring rules** group. The list of rule templates is displayed.
   2. Choose one or more rule templates. When you choose **Save**, WLM creates one rule for each template that you choose.
   3. Enter or confirm values for the rule, including **Rule names**, **Predicates** and **Actions**.
   4. Choose **Save**.

   To add a new rule from scratch, do the following:

   1. To add additional predicates, choose **Add predicate**. You can have up to three predicates for each rule. If all of the predicates are met, WLM triggers the associated action.

2. Choose an **Action**. Each rule has one action.
3. Choose **Save**.

Amazon Redshift generates your WLM configuration parameter in JSON format and displays it in the **JSON** section.

Deleting a parameter group

You can delete a parameter group if you no longer need it and it is not associated with any clusters. You can only delete custom parameter groups.

To delete a parameter group

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Configurations**, then choose **Workload management** to display the **Workload management** page.
3. For **Parameter groups,** choose the parameter group that you want to modify.

> **Note**
>
> You can't delete the default parameter group.

4. Choose **Delete** and confirm that you want to delete the parameter group.

Integrate Amazon Redshift with an AWS Partner

By working with Amazon Redshift, you can integrate with AWS Partners on the Amazon Redshift console. From the **Cluster details** page, you can speed up your data onboarding into your Amazon Redshift data warehouse with AWS Partner applications. You can also join and analyze data from different sources together with existing data in your cluster. Before completing integration with Informatica, you must add the partner's IP addresses to the allowlist of inbound traffic. The following AWS Partners can integrate with Amazon Redshift:

- Datacoral
- Etleap
- Fivetran
- SnapLogic
- Stitch
- Upsolver
- Matillion (preview)
- Sisense (preview)
- Thoughtspot

AWS Partners can integrate with Amazon Redshift using the AWS CLI or Amazon Redshift API operations. For more information, see the *AWS CLI Command Reference* or the *Amazon Redshift API Reference*.

Use the following procedure to integrate a cluster with an AWS Partner.

To integrate an Amazon Redshift cluster with an AWS Partner

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**.
3. Choose the cluster that you want to integrate.
4. Choose **Add partner integration**. The **Choose partner** page opens with details about the available AWS Partners.
5. Choose an AWS Partner, then choose **Next**.

More details about the chosen AWS Partner appear, along with details about the cluster that you are integrating. The **Cluster details** section includes information that you provide on the AWS Partner website such as the **Cluster identifier**, **Endpoint**, **Database name**, and **User name** (which is a database user name). This information is sent to the partner that you chose.

6. Choose **Add partner** to open the AWS Partner's website.
7. Configure the integration with your Amazon Redshift cluster on the partner's website. On the partner's website, you can select and configure the data sources that are loaded to your Amazon Redshift cluster. You can also define additional extract, load, and transform (ELT) transformations to process your business data, join it with other datasets, and build consolidated views for analysis and reporting.

You can view and manage AWS Partner integrations from the cluster details **Properties** tab. The **Integrations** section lists the **Partner** name that you can use to link to the AWS Partner website, the **Status** of the integration, the **Database** that receives the data, and the **Last successful connection** that might have updated the cluster.

The possible status values are as follows:

- Active – The AWS Partner can connect to the cluster and complete configured tasks.
- Inactive – The AWS Partner integration doesn't exist.
- Runtime failure – The AWS Partner can connect to the cluster but can't complete configured tasks.
- Connection failure – The AWS Partner can't connect to the cluster.

After you delete an AWS Partner integration from Amazon Redshift, data continues to flow into your cluster. Complete the delete on the partner's website.

Loading data with AWS partners

Aside from integrating a partner with an Amazon Redshift cluster, you can also move data from more than 30 sources into your Amazon Redshift cluster using our partner's data loading tools. Before you do so, you must add the partner's IP addresses (found below) to the allowlist of inbound rules. For more information about adding rules to an Amazon EC2 security group, see Authorizing Inbound Traffic for Your Instances in the *Amazon EC2 User Guide*. Note that while

the Informatica Data Loader tool is free, data ingress charges might apply depending on the data sources and targets you choose.

You can load data from the following partners:

- Informatica

To load data to an Amazon Redshift cluster with a partner

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose AWS partner integration, then choose the partner you want to integrate your cluster with.
3. Choose **Complete <partner-name> integration**. You will be redirected to the partner's integration site.
4. Enter the necessary details on the partner's site and complete the integration.

Reserved nodes

In AWS, the charges that you accrue for using Amazon Redshift are based on compute nodes. Each compute node is billed at an hourly rate. The hourly rate varies depending on factors such as region, node type, and whether the node receives on-demand node pricing or reserved node pricing.

On-demand node pricing is the most expensive, but most flexible option in Amazon Redshift. With on-demand rates, you are charged only for compute nodes that you have in a running cluster. If you shut down or delete a cluster, you are no longer charged for compute nodes that were in that cluster. You are billed only for the compute nodes that you use, and no more. The hourly rate that you are charged for each compute node varies depending on factors such as region and node type.

Reserved node pricing is less expensive than on-demand pricing because compute nodes are billed at discounted hourly rates. However, to receive these discounted rates, you must purchase reserved node offerings. When you purchase an offering, you make a reservation. The reservation sets a discounted rate for each node that you reserve for the duration of the reservation. The discounted rate in an offering varies depending on factors such as the region, node type, duration, and payment option.

You may designate a node as a reserved node by calling the `PurchaseReservedNodeOffering` API operation or choosing **Purchase reserved nodes** on the Amazon Redshift console. When you purchase a reserved node, you must specify an AWS Region, node type, term, quantity of nodes, and offering type for the applicable reserved node type. The reserved node may only be used in the designated AWS Region.

This topic discusses what reserved node offerings are and how you can purchase them to reduce the cost of running your Amazon Redshift clusters. This topic discusses rates in general terms as on-demand or discounted so you can understand pricing concepts and how pricing affects billing. For more information about specific rates, go to Amazon Redshift Pricing.

Reserved node offerings

If you intend to keep your Amazon Redshift cluster running continuously for a prolonged period, you should consider purchasing reserved node offerings. These offerings provide significant savings over on-demand pricing, but they require you to reserve compute nodes and commit to paying for those nodes for either a one-year or three-year duration.

Reserved nodes are a billing concept that is used strictly to determine the rate at which you are charged for nodes. Reserving a node does not actually create any nodes for you. You are charged for reserved nodes regardless of usage, which means that you must pay for each node that you reserve for the duration of the reservation, whether or not you have any nodes in a running cluster to which the discounted rate applies.

In the evaluation phase of your project or when you're developing a proof of concept, on-demand pricing gives you the flexibility to pay as you go, to pay only for what you use, and to stop paying at any time by shutting down or deleting clusters. After you have established the needs of your production environment and begin the implementation phase, you should consider reserving compute nodes by purchasing one or more offerings.

An offering can apply to one or more compute nodes. You specify the number of compute nodes to reserve when you purchase the offering. You might choose to purchase one offering for multiple compute nodes, or you might choose to purchase multiple offerings and specify a certain number of compute nodes in each offering.

For example, any of the following are valid ways to purchase an offering for three compute nodes:

- Purchase one offering and specify three compute nodes.
- Purchase two offerings, and specify one compute node for the first offering and two compute nodes for the second offering.
- Purchase three offerings, and specify one compute node for each of the offerings.

Comparing pricing among reserved node offerings

Amazon Redshift provides several payment options for offerings. The payment option that you choose affects the payment schedule and the discounted rate that you are charged for the reservation. The more that you pay upfront for the reservation, the better the overall savings are.

The following payment options are available for offerings. The offerings are listed in order from least to most savings over on-demand rates.

> **Note**
>
> You are charged the applicable hourly rate for every hour in the specified duration of the reservation, regardless of whether you use the reserved node or not. The payment

option just determines the frequency of payments and the discount to be applied. For more information, see Reserved node offerings.

Payment option	Payment schedule	Comparative savings	Duration	Upfront charges	Recurring monthly charges
No Upfront	Monthly installments for the duration of the reservation. No upfront payment.	About a 20 percent discount over on-demand rates.	One-year or three-year term	None	Yes
Partial Upfront	Partial upfront payment, and monthly installments for the duration of the reservation.	Up to 41 percent to 73 percent discount depending on duration.	One-year or three-year term	Yes	Yes
All Upfront	Full upfront payment for the reservation. No monthly charges.	Up to 42 percent to 76 percent discount depending on duration.	One-year or three-year term	Yes	None

Specific options and durations are subject to availability.

Note

If you previously purchased **Heavy Utilization** offerings for Amazon Redshift, the comparable offering is the **Partial Upfront** offering.

How reserved nodes work

With reserved node offerings, you pay according to the payment terms as described in the preceding section. You pay this way whether you already have a running cluster or you launch a cluster after you have a reservation.

When you purchase an offering, your reservation has a status of **payment-pending** until the reservation is processed. If the reservation fails to be processed, the status displays as **payment-failed** and you can try the process again. Once your reservation is successfully processed, its status changes to **active**. The applicable discounted rate in your reservation is not applied to your bill until the status changes to **active**. After the reservation duration elapses, the status changes to **retired** but you can continue to access information about the reservation for historical purposes. When a reservation is **retired**, your clusters continue to run but you might be billed at the on-demand rate unless you have another reservation that applies discounted pricing to the nodes.

Reserved nodes are specific to the region in which you purchase the offering. If you purchase an offering by using the Amazon Redshift console, select the AWS region in which you want to purchase an offering, and then complete the reservation process. If you purchase an offering programmatically, the region is determined by the Amazon Redshift endpoint that you connect to. For more information about Amazon Redshift regions, go to Regions and Endpoints in the *Amazon Web Services General Reference*.

To ensure that the discounted rate is applied to all of the nodes when you launch a cluster, make sure that the region, the node type, and the number of nodes that you select match one or more active reservations. Otherwise, you'll be charged at the on-demand rate for nodes that don’t match an active reservation.

In a running cluster, if you exceed the number of nodes that you have reserved, you begin to accrue charges for those additional nodes at the on-demand rate. This accrual means that it is possible for you to be charged varying rates for nodes in the same cluster depending on how many nodes you’ve reserved. You can purchase another offering to cover those additional nodes, and then the discounted rate is applied to those nodes for the remainder of the duration once the reservation status becomes **active**.

If you resize your cluster into a different node type and you haven’t reserved nodes of that type, you’ll be charged at the on-demand rate. You can purchase another offering with the new node type if you want to receive discounted rates for your resized cluster. However, you also continue to pay for the original reservation until its duration elapses. If you need to alter your reservations before the term expires, create a support case using the AWS Console.

Note

The console shows the count of used and unused reserved nodes. However, the console only shows the number of nodes as used that the current user account uses. If another user account under the same payer account uses nodes, the console shows those nodes as unused.

Example

- A payer account reserves 20 nodes
- The current user account uses six nodes
- Another user account under the same payer account also uses six nodes

In this example, the console only displays six used nodes, and fourteen unused nodes.

Reserved nodes and consolidated billing

The pricing benefits of Reserved Nodes are shared when the purchasing account is part of a set of accounts billed under one consolidated billing payer account. The hourly usage across all sub-accounts is aggregated in the payer account every month. This is typically useful for companies in which there are different functional teams or groups; then, the normal Reserved Nodes logic is applied to calculate the bill. For more information, see Consolidated Billing in the AWS Billing User Guide.

Things to consider before changing node type on a cluster that uses Reserved Instances

You can change the node type of a cluster by changing either the node family or the node size using the resize feature (elastic or classic). When you change the node type, the reserved node purchases that are associated with the cluster don't change automatically. The existing reservations remain tied to the original node type. You continue to be billed for these reservations until their term ends. Meanwhile, the nodes of the new type might be billed at the on-demand rate.

Because of this, for changes to your Reserved Instances, we recommend that you contact your AWS account team or AWS Billing Support before you change the node type of a cluster that uses reserved nodes. They can guide you appropriately.

Reserved node examples

The scenarios in this section demonstrate how nodes accrue charges based on on-demand and discounted rates using the following reservation details:

- Region: US West (Oregon)
- Node Type: ra3.xlplus
- Payment Option: No Upfront
- Duration: one year
- Number of Reserved Nodes: 16

Example 1

You have one cluster in the US West (Oregon) region with 20 nodes.

In this scenario, 16 of the nodes receive the discounted rate from the reservation, but the additional 4 nodes in the cluster are billed at the on-demand rate.

Example 2

You have one cluster in the US West (Oregon) region with 12 nodes.

In this scenario, all 12 nodes in the cluster receive the discounted rate from the reservation. However, you also pay for the remaining reserved nodes in the reservation even though you don't currently have a running cluster to which they apply.

Example 3

You have one cluster in the US West (Oregon) region with 12 nodes. You run the cluster for several months with this configuration, and then you need to add nodes to the cluster. You resize the cluster, choosing the same node type and specifying a total of 16 nodes.

In this scenario, you are billed the discounted rate for 16 nodes. Your charges remain the same for the full year duration because the number of nodes that you have in the cluster is equal to the number of nodes that you have reserved.

Example 4

You have one cluster in the US West (Oregon) region with 16 nodes. You run the cluster for several months with this configuration, and then you need to add nodes. You resize the cluster, choosing the same node type and specifying a total of 20 nodes.

In this scenario, you are billed the discounted rate for all the nodes prior to the resize. After the resize, you are billed the discounted rate for 16 of the nodes for the rest of the year, and you are billed at the on-demand rate for the additional 4 nodes that you added to the cluster.

Example 5

You have two clusters in the US West (Oregon) region. One of the clusters has 6 nodes, and the other has 10 nodes.

In this scenario, you are billed at the discounted rate for all of the nodes because the total number of nodes in both clusters is equal to the number of nodes that you have reserved.

Example 6

You have two clusters in the US West (Oregon) region. One of the clusters has 4 nodes, and the other has 6 nodes.

In this scenario, you are billed the discounted rate for the 10 nodes that you have in running clusters, and you also pay the discounted rate for the additional 6 nodes that you have reserved even though you don't currently have any running clusters to which they apply.

Purchasing a reserved node

You can use the AWS Management Console or the AWS CLI to purchase reserved node offerings, and to view current and past reservations.

AWS Management Console

To purchase a reserved node

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose **Reserved nodes** to display the list of reserved nodes.

3. Choose **Purchase reserved nodes** to display the page to choose the properties of the node that you want to purchase.
4. Enter the properties of the node, then choose **Purchase reserved nodes**.

After you purchase an offering, the **Reserved Node** list displays your reservations and the details of each one, such as the node type, number of nodes, and status of the reservation. For more information about the reservation details, see How reserved nodes work.

To upgrade a reserved node, use the AWS CLI.

You can't convert all node types to reserved nodes, and it's also possible that an existing reserved node isn't available for renewal. This might be because the node type is discontinued. Contact customer support to renew a discontinued node type.

AWS CLI

To upgrade a reserved node reservation with the AWS CLI

1. Obtain a list of ReservedNodeOfferingID's for offerings that meet your requirements for payment type, term, and charges. The following example illustrates this step.

```
aws redshift get-reserved-node-exchange-offerings --reserved-node-id xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxxxx
{
    "ReservedNodeOfferings": [
        {
            "Duration": 31536000,
            "ReservedNodeOfferingId": "yyyyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy",
            "UsagePrice": 0.0,
            "NodeType": "dc2.large",
            "RecurringCharges": [
                {
                    "RecurringChargeFrequency": "Hourly",
                    "RecurringChargeAmount": 0.2
                }
            ],
            "CurrencyCode": "USD",
            "OfferingType": "No Upfront",
            "ReservedNodeOfferingType": "Regular",
            "FixedPrice": 0.0
        }
```

```
    ]
}
```

2. Call `accept-reserved-node-exchange` and provide the ID for the DC1 reserved node that you want to exchange along with the ReservedNodeOfferingID you obtained in the previous step.

   The following example illustrates this step.

```
aws redshift accept-reserved-node-exchange --reserved-node-id xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxx --target-reserved-node-offering-id yyyyyyyy-yyyy-yyyy-
yyyy-yyyyyyyyyyyy
{
    "ExchangedReservedNode": {
        "UsagePrice": 0.0,
        "OfferingType": "No Upfront",
        "State": "exchanging",
        "FixedPrice": 0.0,
        "CurrencyCode": "USD",
        "ReservedNodeId": "zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzzzzzz",
        "NodeType": "dc2.large",
        "NodeCount": 1,
        "RecurringCharges": [
            {
                "RecurringChargeFrequency": "Hourly",
                "RecurringChargeAmount": 0.2
            }
        ],
        "ReservedNodeOfferingType": "Regular",
        "StartTime": "2018-06-27T18:02:58Z",
        "ReservedNodeOfferingId": "yyyyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy",
        "Duration": 31536000
    }
}
```

You can confirm that the exchange is complete by calling describe-reserved-nodes and checking the value for `Node type`.

Security in Amazon Redshift

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. The effectiveness of our security is regularly tested and verified by third-party auditors as part of the AWS compliance programs. To learn about the compliance programs that apply to Amazon Redshift, see AWS services in scope by compliance program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations.

Access to Amazon Redshift resources is controlled at four levels:

- **Cluster management –** The ability to create, configure, and delete clusters is controlled by the permissions given to the user or account associated with your AWS security credentials. Users with the proper permissions can use the AWS Management Console, AWS Command Line Interface (CLI), or Amazon Redshift Application Programming Interface (API) to manage their clusters. This access is managed by using IAM policies.

  > **Important**
  >
  > Amazon Redshift has a collection of best practices for managing permissions, identities and secure access. We recommend that you get familiar with these as you get started with Amazon Redshift. For more information, see Identity and access management in Amazon Redshift.

- **Cluster connectivity –** Amazon Redshift security groups specify the AWS instances that are authorized to connect to an Amazon Redshift cluster in Classless Inter-Domain Routing (CIDR) format. For information about creating Amazon Redshift, Amazon EC2, and Amazon VPC security groups and associating them with clusters, see Amazon Redshift security groups.

- **Database access –** The ability to access database objects, such as tables and views, is controlled by database user accounts in the Amazon Redshift database. Users can only access resources in the database that their user accounts have been granted permission to access. You create these Amazon Redshift user accounts and manage permissions by using the CREATE USER, CREATE GROUP, GRANT, and REVOKE SQL statements. For more information, see Managing database security in the Amazon Redshift Database Developer Guide.
- **Temporary database credentials and single sign-on –** In addition to creating and managing database users using SQL commands, such as CREATE USER and ALTER USER, you can configure your SQL client with custom Amazon Redshift JDBC or ODBC drivers. These drivers manage the process of creating database users and temporary passwords as part of the database logon process.

  The drivers authenticate database users based on AWS Identity and Access Management (IAM) authentication. If you already manage user identities outside of AWS, you can use a SAML 2.0-compliant identity provider (IdP) to manage access to Amazon Redshift resources. You use an IAM role to configure your IdP and AWS to permit your federated users to generate temporary database credentials and log on to Amazon Redshift databases. For more information, see Using IAM authentication to generate database user credentials.

This documentation helps you understand how to apply the shared responsibility model when using Amazon Redshift. The following topics show you how to configure Amazon Redshift to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon Redshift resources.

Topics

- Data protection in Amazon Redshift
- Identity and access management in Amazon Redshift
- Managing Amazon Redshift admin passwords using AWS Secrets Manager
- Logging and monitoring in Amazon Redshift
- Compliance validation for Amazon Redshift
- Resilience in Amazon Redshift
- Infrastructure security in Amazon Redshift
- Configuration and vulnerability analysis in Amazon Redshift

Data protection in Amazon Redshift

The AWS shared responsibility model applies to data protection in Amazon Redshift. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see Data Privacy FAQ. For information about data protection in Europe, see the General Data Protection Regulation (GDPR) Center.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with Amazon Redshift or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Data encryption

Data protection refers to protecting data while in transit (as it travels to and from Amazon Redshift) and at rest (while it is stored on disks in Amazon Redshift data centers). You can protect data in transit by using SSL or by using client-side encryption. You have the following options of protecting data at rest in Amazon Redshift.

- **Use server-side encryption** – You request Amazon Redshift to encrypt your data before saving it on disks in its data centers and decrypt it when you download the objects.
- **Use client-side encryption** – You can encrypt data client-side and upload the encrypted data to Amazon Redshift. In this case, you manage the encryption process, the encryption keys, and related tools.

Encryption at rest

Server-side encryption is about data encryption at rest—that is, Amazon Redshift optionally encrypts your data as it writes it in its data centers and decrypts it for you when you access it. As long as you authenticate your request and you have access permissions, there is no difference in the way you access encrypted or unencrypted data.

Amazon Redshift protects data at rest through encryption. Optionally, you can protect all data stored on disks within a cluster and all backups in Amazon S3 with Advanced Encryption Standard AES-256.

To manage the keys used for encrypting and decrypting your Amazon Redshift resources, you use AWS Key Management Service (AWS KMS). AWS KMS combines secure, highly available hardware and software to provide a key management system scaled for the cloud. Using AWS KMS, you can create encryption keys and define the policies that control how these keys can be used. AWS KMS supports AWS CloudTrail, so you can audit key usage to verify that keys are being used appropriately. You can use your AWS KMS keys in combination with Amazon Redshift and supported AWS services.. For a list of services that support AWS KMS, see How AWS Services Use AWS KMS in the *AWS Key Management Service Developer Guide*.

If you choose to manage your provisioned cluster or serverless namespace's admin password using AWS Secrets Manager, Amazon Redshift also accepts an additional AWS KMS key that AWS Secrets Manager uses to encrypt your credentials. This additional key can be an automatically generated key from AWS Secrets Manager, or a custom key that you provide.

Amazon Redshift query editor v2 securely stores information entered into the query editor as follows:

- The Amazon Resource Name (ARN) of the KMS key used to encrypt query editor v2 data.
- Database connection information.
- Names and content of files and folders.

Amazon Redshift query editor v2 encrypts information using block-level encryption with either your KMS key or the service account KMS key. The encryption of your Amazon Redshift data is controlled by your Amazon Redshift cluster properties.

Topics

- Amazon Redshift database encryption

Amazon Redshift database encryption

In Amazon Redshift, your database is encrypted by default to protect your data at rest. Database encryption applies to the cluster and also to its snapshots.

You can modify an unencrypted cluster to use AWS Key Management Service (AWS KMS) encryption. To do so, you can use either an AWS-owned key or a customer managed key. When you modify your cluster to enable AWS KMS encryption, Amazon Redshift automatically migrates your data to a new encrypted cluster. Snapshots created from the encrypted cluster are also encrypted. You can also migrate an encrypted cluster to an unencrypted cluster by modifying the cluster and changing the **Encrypt database** option. For more information, see Changing cluster encryption.

Though you can still transform the default encrypted cluster to unencrypted after creating the cluster, we recommend that you keep cluster that contains sensitive data as encrypted. 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.

Amazon Redshift uses a hierarchy of encryption keys to encrypt the database. You can use either AWS Key Management Service (AWS KMS) or a hardware security module (HSM) to manage the top-level encryption keys in this hierarchy. The process that Amazon Redshift uses for encryption differs depending on how you manage keys. Amazon Redshift automatically integrates with AWS

KMS but not with an HSM. When you use an HSM, you must use client and server certificates to configure a trusted connection between Amazon Redshift and your HSM.

Important

Amazon Redshift can lose access to the KMS key for a provisioned cluster or serverless namespace when you disable the customer-managed KMS key. In these cases, Amazon Redshift takes a backup of the Amazon Redshift data warehouse and puts it into an `inaccessible-kms-key` state for 14 days. If you restore the KMS key within that period, Amazon Redshift will restore access and the warehouse will function normally. If the 14 day period ends without the KMS key being restored, Amazon Redshift will delete the data warehouse. While a warehouse is in `inaccessible-kms-key` state, it has the following characteristics:

- You can't run any queries on the data warehouse.
- If the data warehouse is the producer warehouse of a datashare, you can't run datasharing queries against it from consumer warehouses.
- You can't create cross-region snapshot copies.

For information on restoring a disabled KMS key, see Enable and disable keys in the *AWS Key Management Service Developer Guide*. If the warehouse's KMS key was deleted, you can use the backup to create a new data warehouse before the `inaccessible-kms-key` status warehouse is deleted.

Encryption process improvements for better performance and availability

Encryption with RG and RA3 nodes

Updates to the encryption process for RG and RA3 nodes have made the experience much better. Both read and write queries can run during the process with less performance impact from the encryption. Also, encryption finishes much more quickly. The updated process steps include a restore operation and migration of cluster metadata to a target cluster. The improved experience applies to encryption types like AWS KMS, for example. When you have petabyte-scale data volumes, the operation has been reduced from weeks to days.

Prior to encrypting your cluster, if you plan to continue to run database workloads, you can improve performance and speed up the process by adding nodes with elastic resize. You can't use

elastic resize when encryption is in process, so do it before you encrypt. Note that adding nodes typically results in higher cost.

Encryption with other node types

When you encrypt a cluster with DC2 nodes, you don't have the ability to run write queries, like with RG or RA3 nodes. Only read queries can be run.

Usage notes for encryption with RG and RA3 nodes

The following insights and resources help you prepare for encryption and monitor the process.

- **Running queries after starting encryption** – After encryption is started, reads and writes are available within about fifteen minutes. How long it takes the full encryption process to complete depends on the amount of data on the cluster and the workload levels.
- **How long does encryption take?** – The time to encrypt your data depends on several factors: These include the number of workloads running, the compute resources being used, the number of nodes, and the type of nodes. We recommend that you initially perform encryption in a test environment. As a rule of thumb, if you're working with data volumes in petabytes, it likely can take 1-3 days for encryption to complete.
- **How do I know encryption is finished?** – After you enable encryption, the completion of the first snapshot confirms that encryption is completed.
- **Rolling back encryption** – If you need to roll back the encryption operation, the best way to do this is to restore from the most recent backup taken prior to when encryption was initiated. You will have to re-apply any new updates (updates/deletes/inserts) following the last-backup.
- **Performing a table restore** – Note that you can't restore a table from an unencrypted cluster to an encrypted cluster.
- **Encrypting a single-node cluster** – Encrypting a single-node cluster has performance limitations. It takes longer than encryption for a multi-node cluster.
- **Creating a backup after encryption** – When you encrypt the data in your cluster, a backup isn't created until the cluster is fully encrypted. The amount of time this takes can vary. The time taken for backup can be hours to days, depending on the cluster size. After encryption completes, there can be a delay before you can create a backup.

  Note that because a backup-and-restore operation occurs during the encryption process, any tables or materialized views created with BACKUP NO aren't retained. For more information, see CREATE TABLE or CREATE MATERIALIZED VIEW.

Topics

- Encryption using AWS KMS
- Encryption using hardware security modules
- Encryption key rotation
- Changing cluster encryption
- Migrating to an HSM-encrypted cluster
- Rotating encryption keys

Encryption using AWS KMS

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 root 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 AWS KMS keys that Amazon Redshift or your AWS account has created or has permission to use in AWS KMS. You select a KMS key to use as your root key in the encryption hierarchy.

By default, Amazon Redshift selects an automatically generated AWS-owned key as the root key for your AWS account to use in Amazon Redshift.

If you don't want to use the default key, you must have (or create) a customer managed KMS key separately in AWS KMS before you launch your cluster in Amazon Redshift. Customer managed keys 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 KMS keys, see Creating Keys in the *AWS Key Management Service Developer Guide*.

If you want to use a AWS KMS key from another AWS account, you must have permission to use the key and specify its Amazon Resource Name (ARN) in Amazon Redshift. For more information about access to keys in AWS KMS, see Controlling Access to Your Keys in the *AWS Key Management Service Developer Guide*.

After you choose a root key, Amazon Redshift requests that AWS KMS generate a data key and encrypt it using the selected root 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 KMS key and the encryption context for the CEK. Only the encrypted CEK is exported to Amazon Redshift; the KMS key 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. For more information about grants, encryption context, and other AWS KMS-related concepts, see Concepts in the *AWS Key Management Service Developer Guide*.

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 KMS key 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.

For more information about creating Amazon Redshift clusters that are encrypted with AWS KMS keys, see Creating a cluster.

Copying AWS KMS–encrypted snapshots to another AWS Region

AWS KMS keys are specific to an AWS Region. If you want to enable copying of Amazon Redshift snapshots from an encrypted source cluster to another AWS Region, but want to use your own AWS KMS key for snapshots in the destination, you need to configure a grant for Amazon Redshift to use a root key in your account in the destination AWS Region. This grant enables Amazon Redshift to encrypt snapshots in the destination AWS Region. If you want snapshots in destination encrypted by an AWS Region-owned key, you don't need to configure any grants in the destination AWS Region. For more information about cross-Region snapshot copy, see Copying a snapshot to another AWS Region.

> **Note**
>
> If you enable copying of snapshots from an encrypted cluster and use AWS KMS for your root key, you cannot rename your cluster because the cluster name is part of the encryption context. If you must rename your cluster, you can disable copying of snapshots in the source AWS Region, rename the cluster, and then configure and enable copying of snapshots again.

The process to configure the grant for copying snapshots is as follows.

1. In the destination AWS Region, create a snapshot copy grant by doing the following:
   - If you do not already have an AWS KMS key to use, create one. For more information about creating AWS KMS keys, see Creating Keys in the *AWS Key Management Service Developer Guide*.
   - Specify a name for the snapshot copy grant. This name must be unique in that AWS Region for your AWS account.
   - Specify the AWS KMS key ID for which you are creating the grant. If you do not specify a key ID, the grant applies to your default key.
2. In the source AWS Region, enable copying of snapshots and specify the name of the snapshot copy grant that you created in the destination AWS Region.

This preceding process is only necessary if you enable copying of snapshots using the AWS CLI, the Amazon Redshift API, or SDKs. If you use the console, Amazon Redshift provides the proper workflow to configure the grant when you enable cross-Region snapshot copy. For more information about configuring cross-Region snapshot copy for AWS KMS-encrypted clusters by using the console, see Configuring cross-Region snapshot copy for an AWS KMS–encrypted cluster.

Before the snapshot is copied to the destination AWS Region, Amazon Redshift decrypts the snapshot using the root key in the source AWS Region and re-encrypts it temporarily using a randomly generated RSA key that Amazon Redshift manages internally. Amazon Redshift then copies the snapshot over a secure channel to the destination AWS Region, decrypts the snapshot using the internally managed RSA key, and then re-encrypts the snapshot using the root key in the destination AWS Region.

Encryption using hardware security modules

If you don't use AWS KMS for key management, you can use a hardware security module (HSM) for key management with Amazon Redshift.

> **Important**
>
> HSM encryption is not supported for DC2, RG, and RA3 node types.

HSMs are devices that provide direct control of key generation and management. They provide greater security by separating key management from the application and database layers. Amazon

Redshift supports AWS CloudHSM Classic for key management. The encryption process is different when you use HSM to manage your encryption keys instead of AWS KMS.

> **Important**
>
> Amazon Redshift supports only AWS CloudHSM Classic. We don't support the newer AWS CloudHSM service.
> AWS CloudHSM Classic is closed to new customers. For more information, see CloudHSM Classic Pricing. AWS CloudHSM Classic isn't available in all AWS Regions. For more information about available AWS Regions, see AWS Region Table.

When you configure your cluster to use an HSM, Amazon Redshift sends a request to the HSM to generate and store a key to be used as the CEK. However, unlike AWS KMS, the HSM doesn't export the CEK to Amazon Redshift. Instead, Amazon Redshift randomly generates the DEK in the cluster and passes it to the HSM to be encrypted by the CEK. The HSM returns the encrypted DEK to Amazon Redshift, where it is further encrypted using a randomly-generated, internal root key and stored internally on disk in a separate network from the cluster. Amazon Redshift also loads the decrypted version of the DEK in memory in the cluster so that the DEK can be used to encrypt and decrypt the individual keys for the data blocks.

If the cluster is rebooted, Amazon Redshift decrypts the internally-stored, double-encrypted DEK using the internal root key to return the internally stored DEK to the CEK-encrypted state. The CEK-encrypted DEK is then passed to the HSM to be decrypted and passed back to Amazon Redshift, where it can be loaded in memory again for use with the individual data block keys.

Configuring a trusted connection between Amazon Redshift and an HSM

When you opt to use an HSM for management of your cluster key, you need to configure a trusted network link between Amazon Redshift and your HSM. Doing this requires configuration of client and server certificates. The trusted connection is used to pass the encryption keys between the HSM and Amazon Redshift during encryption and decryption operations.

Amazon Redshift creates a public client certificate from a randomly generated private and public key pair. These are encrypted and stored internally. You download and register the public client certificate in your HSM, and assign it to the applicable HSM partition.

You provide Amazon Redshift with the HSM IP address, HSM partition name, HSM partition password, and a public HSM server certificate, which is encrypted by using an internal root key.

Amazon Redshift completes the configuration process and verifies that it can connect to the HSM. If it cannot, the cluster is put into the INCOMPATIBLE_HSM state and the cluster is not created. In this case, you must delete the incomplete cluster and try again.

> **Important**
>
> When you modify your cluster to use a different HSM partition, Amazon Redshift verifies that it can connect to the new partition, but it does not verify that a valid encryption key exists. Before you use the new partition, you must replicate your keys to the new partition. If the cluster is restarted and Amazon Redshift cannot find a valid key, the restart fails. For more information, see Replicating Keys Across HSMs.

After initial configuration, if Amazon Redshift fails to connect to the HSM, an event is logged. For more information about these events, see Amazon Redshift Event Notifications.

Encryption key rotation

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.

> **Note**
>
> You cannot rotate keys for snapshots without a source cluster. Before you delete a cluster, consider whether its snapshots rely on key rotation.

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.

For more information about rotating keys, see Rotating encryption keys.

Changing cluster encryption

You can modify an unencrypted cluster to use AWS Key Management Service (AWS KMS) encryption using either an AWS-owned key or a customer managed key. When you modify your cluster to enable AWS KMS encryption, Amazon Redshift automatically migrates your data to a new encrypted cluster. You can also migrate an encrypted cluster to an unencrypted cluster by modifying the cluster with the AWS CLI, but not with the AWS Management Console.

During the migration operation, your cluster is available in read-only mode, and the cluster status appears as **resizing**.

If your cluster is configured to enable cross-AWS Region snapshot copy, you must disable it before changing encryption. For more information, see Copying a snapshot to another AWS Region and Configuring cross-Region snapshot copy for an AWS KMS–encrypted cluster. You can't enable hardware security module (HSM) encryption by modifying the cluster. Instead, create a new, HSM-encrypted cluster and migrate your data to the new cluster. For more information, see Migrating to an HSM-encrypted cluster.

Amazon Redshift console

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the cluster that you want to modify encryption.
3. Choose **Properties**.
4. In the **Database configurations** section, choose **Edit**, then choose **Edit encryption**.
5. Choose one of the encryption options and choose **Save changes**.

AWS CLI

To modify your unencrypted cluster to use AWS KMS, run the `modify-cluster` CLI command and specify `--encrypted`, as shown following. By default, your default KMS key is used. To specify a customer managed key, include the `--kms-key-id` option.

```
aws redshift modify-cluster --cluster-identifier <value> --encrypted --kms-key-id
 <value>
```

To remove encryption from your cluster, run the following CLI command.

```
aws redshift modify-cluster --cluster-identifier <value> --no-encrypted
```

Migrating to an HSM-encrypted cluster

To migrate an unencrypted cluster to a cluster encrypted using a hardware security module (HSM), you create a new encrypted cluster and move your data to the new cluster. You can't migrate to an HSM-encrypted cluster by modifying the cluster.

To migrate from an unencrypted cluster to an HSM-encrypted cluster, you first unload your data from the existing, source cluster. Then you reload the data in a new, target cluster with the chosen encryption setting. For more information about launching an encrypted cluster, see Amazon Redshift database encryption.

During the migration process, your source cluster is available for read-only queries until the last step. The last step is to rename the target and source clusters, which switches endpoints so all traffic is routed to the new, target cluster. The target cluster is unavailable until you reboot following the rename. Suspend all data loads and other write operations on the source cluster while data is being transferred.

To prepare for migration

1. Identify all the dependent systems that interact with Amazon Redshift, for example business intelligence (BI) tools and extract, transform, and load (ETL) systems.
2. Identify validation queries to test the migration.

   For example, you can use the following query to find the number of user-defined tables.

   ```
   select count(*)
   from pg_table_def
   where schemaname != 'pg_catalog';
   ```

   The following query returns a list of all user-defined tables and the number of rows in each table.

```
select "table", tbl_rows
from svv_table_info;
```

3. Choose a good time for your migration. To find a time when cluster usage is lowest, monitor cluster metrics such as CPU utilization and number of database connections. For more information, see Viewing cluster performance data.
4. Drop unused tables.

   To create a list of tables and the number of the times each table has been queried, run the following query.

```
select database,
schema,
table_id,
"table",
round(size::float/(1024*1024)::float,2) as size,
sortkey1,
nvl(s.num_qs,0) num_qs
from svv_table_info t
left join (select tbl,
perm_table_name,
count(distinct query) num_qs
from stl_scan s
where s.userid > 1
and   s.perm_table_name not in ('Internal worktable','S3')
group by tbl,
perm_table_name) s on s.tbl = t.table_id
where t."schema" not in ('pg_internal');
```

5. Launch a new, encrypted cluster.

   Use the same port number for the target cluster as for the source cluster. For more information about launching an encrypted cluster, see Amazon Redshift database encryption.
6. Set up the unload and load process.

   You can use the Amazon Redshift Unload/Copy Utility to help you to migrate data between clusters. The utility exports data from the source cluster to a location on Amazon S3. The data is encrypted with AWS KMS. The utility then automatically imports the data into the target. Optionally, you can use the utility to clean up Amazon S3 after migration is complete.
7. Run a test to verify your process and estimate how long write operations must be suspended.

During the unload and load operations, maintain data consistency by suspending data loads and other write operations. Using one of your largest tables, run through the unload and load process to help you estimate timing.

8. Create database objects, such as schemas, views, and tables. To help you generate the necessary data definition language (DDL) statements, you can use the scripts in AdminViews in the AWS GitHub repository.

To migrate your cluster

1. Stop all ETL processes on the source cluster.

   To confirm that there are no write operations in process, use the Amazon Redshift Management Console to monitor write IOPS. For more information, see Viewing cluster performance data.

2. Run the validation queries you identified earlier to collect information about the unencrypted source cluster before migration.

3. (Optional) Create one workload management (WLM) queue to use the maximum available resources in both the source and target cluster. For example, create a queue named `data_migrate` and configure the queue with memory of 95 percent and concurrency of 4. For more information, see Routing Queries to Queues Based on User Groups and Query Groups in the *Amazon Redshift Database Developer Guide*.

4. Using the `data_migrate` queue, run the UnloadCopyUtility.

   Monitor the UNLOAD and COPY process using the Amazon Redshift Console.

5. Run the validation queries again and verify that the results match the results from the source cluster.

6. Rename your source and target clusters to swap the endpoints. To avoid disruption, perform this operation outside of business hours.

7. Verify that you can connect to the target cluster using all of your SQL clients, such as ETL and reporting tools.

8. Shut down the unencrypted source cluster.

Rotating encryption keys

You can use the following procedure to rotate encryption keys by using the Amazon Redshift console.

To rotate the encryption keys for a cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the cluster that you want to update encryption keys.
3. For **Actions**, choose **Rotate encryption** to display the **Rotate encryption keys** page.
4. On the **Rotate encryption keys** page, choose **Rotate encryption keys**.

Encryption in transit

You can configure your environment to protect the confidentiality and integrity of data in transit.

The following details apply to encryption of data in transit between an Amazon Redshift cluster and SQL clients over JDBC/ODBC:

- You can connect to Amazon Redshift clusters from SQL client tools over Java Database Connectivity (JDBC) and Open Database Connectivity (ODBC) connections.
- Amazon Redshift supports Secure Sockets Layer (SSL) connections to encrypt data and server certificates to validate the server certificate that the client connects to. The client connects to the leader node of an Amazon Redshift cluster. For more information, see Configuring security options for connections.
- To support SSL connections, Amazon Redshift creates and installs AWS Certificate Manager (ACM) issued certificates on each cluster. For more information, see Transitioning to ACM certificates for SSL connections.
- To protect your data in transit within the AWS Cloud, Amazon Redshift uses hardware accelerated SSL to communicate with Amazon S3 or Amazon DynamoDB for COPY, UNLOAD, backup, and restore operations.

The following details apply to encryption of data in transit between an Amazon Redshift cluster and Amazon S3 or DynamoDB:

- Amazon Redshift uses hardware accelerated SSL to communicate with Amazon S3 or DynamoDB for COPY, UNLOAD, backup, and restore operations.
- Redshift Spectrum supports the Amazon S3 server-side encryption (SSE) using your account's default key managed by the AWS Key Management Service (KMS).
- You can encrypt Amazon Redshift loads with Amazon S3 and AWS KMS. For more information, see Encrypt Your Amazon Redshift Loads with Amazon S3 and AWS KMS.

The following details apply to encryption and signing of data in transit between AWS CLI, SDK, or API clients and Amazon Redshift endpoints:

- Amazon Redshift provides HTTPS endpoints for encrypting data in transit.
- To protect the integrity of API requests to Amazon Redshift, API calls must be signed by the caller. Calls are signed by an X.509 certificate or the customer's AWS secret access key according to the Signature Version 4 Signing Process (Sigv4). For more information, see Signature Version 4 Signing Process in the *AWS General Reference*.
- Use the AWS CLI or one of the AWS SDKs to make requests to AWS. These tools automatically sign the requests for you with the access key that you specify when you configure the tools.

The following details apply to encryption of data in transit between Amazon Redshift clusters and Amazon Redshift query editor v2:

- Data is transmitted between query editor v2 and Amazon Redshift clusters over a TLS-encrypted channel.

VPC encryption controls with Amazon Redshift

Amazon Redshift supports VPC encryption controls, a security feature that helps you enforce encryption in transit for all traffic within and across VPCs in a Region. This document describes how to use VPC encryption controls with Amazon Redshift clusters and serverless workgroups.

VPC encryption controls provide centralized control to monitor and enforce encryption in transit within your VPCs. When enabled in enforce mode, it ensures that all network traffic is encrypted either at the hardware layer (using AWS Nitro System) or at the application layer (using TLS/SSL).

Amazon Redshift integrates with VPC encryption controls to help you meet compliance requirements for industries such as healthcare (HIPAA), government (FedRAMP), and finance (PCI DSS).

How VPC encryption controls work with Amazon Redshift

VPC encryption controls operate in two modes:

- Monitor Mode: Provides visibility into the encryption status of traffic flows and helps identify resources that allow unencrypted traffic.
- Enforce Mode: Prevents the creation or use of resources that allow unencrypted traffic within the VPC. All traffic must be encrypted either at the hardware layer (Nitro-based instances) or application layer (TLS/SSL).

Requirements for using VPC encryption controls

Instance type requirements

Amazon Redshift requires Nitro-based instances to support VPC encryption controls. All modern Redshift instance types support the necessary encryption capabilities.

SSL/TLS requirements

When VPC encryption controls is enabled in enforce mode, the require_ssl parameter must be set to true and cannot be disabled. This ensures that all client connections use encrypted TLS connections.

Migrating to VPC encryption controls

For existing clusters and workgroups

You cannot enable VPC encryption controls in enforce mode on a VPC that contains existing Redshift clusters or serverless workgroups. See the following steps to use encryption controls if you have an existing cluster or workgroup:

1. Create a snapshot of your existing cluster or namespace
2. Create a new VPC with VPC encryption controls enabled in enforce mode
3. Restore from the snapshot into the new VPC using one of these operations:
    - For provisioned clusters: Use the `restore-from-cluster-snapshot` operation
    - For serverless: Use the `restore-from-snapshot` operation on your workgroup

When creating new clusters or workgroups in a VPC with encryption controls enabled, the require_ssl parameter must be set to true.

Amazon Redshift requires Nitro-based instances to support VPC encryption controls. All modern Redshift instance types support the necessary encryption capabilities.

SSL/TLS requirements

When VPC encryption controls is enabled in enforce mode, the require_ssl parameter must be set to true and cannot be disabled. This ensures that all client connections use encrypted TLS connections.

Considerations and limitations

When using VPC encryption controls in Amazon Redshift, consider the following:

VPC State Restrictions

- Cluster and workgroup creation is blocked when VPC encryption controls is in `enforce-in-progress` state
- You must wait until the VPC reaches `enforce` mode before creating new resources

SSL configuration

- require_ssl parameter: Must always be `true` for clusters and workgroups created in encryption-enforced VPCs
- Once a cluster or workgroup is created in an encryption-enforced VPC, `require_ssl` cannot be disabled for its lifetime

Region availability

This feature is not available in enforce mode with Amazon Redshift Serverless in the following Regions:

- South America (São Paulo)
- Europe (Zurich)

Key management

You can configure your environment to protect data with keys:

- Amazon Redshift automatically integrates with AWS Key Management Service (AWS KMS) for key management. AWS KMS uses envelope encryption. For more information, see Envelope Encryption.
- When encryption keys are managed in AWS KMS, Amazon Redshift uses a four-tier, key-based architecture for encryption. The architecture consists of randomly generated AES-256 data encryption keys, a database key, a cluster key, and a root key. For more information, see How Amazon Redshift Uses AWS KMS.
- You can create your own customer managed key in AWS KMS. For more information, see Create a KMS key and Authorizing use of an AWS KMS customer managed key with Amazon Redshift (Provisioned) and Amazon Redshift Serverless.
- You can also import your own key material for new AWS KMS keys. For more information, see Importing Key Material in AWS Key Management Service (AWS KMS).
- Amazon Redshift supports management of encryption keys in external hardware security modules (HSMs). The HSM can be on-premises or can be AWS CloudHSM. When you use an HSM, you must use client and server certificates to configure a trusted connection between Amazon Redshift and your HSM. Amazon Redshift supports only AWS CloudHSM Classic for key management. For more information, see Encryption using hardware security modules. For information about AWS CloudHSM, see What is AWS CloudHSM?
- You can rotate encryption keys for encrypted clusters.. For more information, see Encryption key rotation.

Authorizing use of an AWS KMS customer managed key with Amazon Redshift (Provisioned) and Amazon Redshift Serverless

Amazon Redshift integrates with AWS Key Management Service (AWS KMS) to encrypt your data at rest. When you use a customer managed key, you have full control over the key policy, key rotation, and enabling or disabling the key. For more information about customer managed keys, see Customer managed keys in the *AWS Key Management Service Developer Guide*.

When Amazon Redshift uses a customer managed key in cryptographic operations, it acts on behalf of the user who is creating or modifying the Amazon Redshift provisioned cluster or Amazon Redshift Serverless resource.

Required permissions

To create and use an Amazon Redshift provisioned cluster or an Amazon Redshift Serverless resource that is encrypted with a customer managed key, a user must have permission to call the following operations on the customer managed key:

- `kms:CreateGrant`
- `kms:GenerateDataKey`
- `kms:Encrypt`
- `kms:Decrypt`
- `kms:DescribeKey`

The owner or administrator of the customer managed key can add these permissions for specific users in the key policy of the customer managed key, or in an IAM policy if the key policy allows it.

Amazon Redshift accesses the customer managed key through AWS KMS grants. Make sure that your key policy doesn't prevent Amazon Redshift from creating or using grants on the key. For example, explicit Deny statements, or restrictive conditions on Allow statements, can block grant creation and cause operations to fail.

As a security best practice, we recommend that you provide only the permissions that are required for your use case. For example, you can scope down the `kms:CreateGrant` permission by using a condition key such as `kms:GrantIsForAWSResource`.

For more information about controlling access to AWS KMS keys, see Key policies in AWS KMS and Allowing users in other accounts to use a KMS key in the *AWS Key Management Service Developer Guide*.

Data tokenization

Tokenization is the process of replacing actual values with opaque values for data security purposes. Security-sensitive applications use tokenization to replace sensitive data such as personally identifiable information (PII) or protected health information (PHI) with tokens to reduce the security risks. *Detokenization* reverses tokens with actual values for authorized users with appropriate security policies.

For integration with third-party tokenization services, you can use Amazon Redshift user-defined functions (UDFs) that you create using AWS Lambda. For more information, see Lambda user-defined functions in the *Amazon Redshift Database Developer Guide*. For example, see Protegrity.

Amazon Redshift sends tokenization requests to a tokenization server accessed through a REST API or predefined endpoint. Two or more complimentary Lambda functions process the tokenization and detokenization requests. For this processing, you can use Lambda functions provided by a third-party tokenization provider. You can also use Lambda functions that you register as Lambda UDFs in Amazon Redshift.

For example, suppose that a query is submitted that invokes a tokenization or detokenization UDF on a column. The Amazon Redshift cluster spools the applicable rows of arguments and sends those rows in batches to the Lambda function in parallel. The data transfers between the Amazon Redshift compute nodes and Lambda in a separate, isolated network connection that's not accessible to clients. The Lambda function passes the data to the tokenization server endpoint. The tokenization server tokenizes or detokenizes the data as necessary and returns it. The Lambda functions then transmit the results to the Amazon Redshift cluster for further processing, if necessary, and then return the query results.

Routing internetwork traffic in Amazon Redshift

You can route traffic through known and private network routes in Amazon Redshift. This page covers how to route traffic on a corporate network and between resources in the same AWS Region.

To route traffic between Amazon Redshift and clients and applications on a corporate network:

- Set up a private connection between your virtual private cloud (VPC) and your corporate network. Set up either an IPsec VPN connection over the internet or a private physical connection using Direct Connect connection. Direct Connect enables you to establish a private virtual interface from your on-premises network directly to your Amazon VPC, providing you with a private, high-bandwidth network connection between your network and your VPC. With multiple virtual interfaces, you can even establish private connectivity to multiple VPCs while maintaining network isolation. For more information, see What is AWS Site-to-Site VPN? and What is Direct Connect?

To route traffic between an Amazon Redshift cluster in a VPC and Amazon S3 buckets in the same AWS Region:

- Set up an Amazon S3 private VPC endpoint to privately access Amazon S3 data from an ETL load or unload. For more information, see Endpoints for Amazon S3.
- Enable “Enhanced VPC routing” for an Amazon Redshift cluster, specifying a target Amazon S3 VPC endpoint. Traffic generated by Amazon Redshift COPY, UNLOAD, or CREATE LIBRARY

commands are then routed through the private endpoint. For more information, see Turning on enhanced VPC routing.

Identity and access management in Amazon Redshift

Access to Amazon Redshift requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access AWS resources, such as an Amazon Redshift cluster. The following sections provide details on how you can use AWS Identity and Access Management (IAM) and Amazon Redshift to help secure your resources by controlling who can access them:

- Authentication with identities
- Access control

> **Important**
>
> This topic contains a collection of best practices for managing permissions, identities and secure access. We recommend that you get familiar with best practices for using IAM with Amazon Redshift. These include using IAM roles for applying permissions. Having a good understanding of these sections will help you maintain a more secure Amazon Redshift data warehouse.

Authentication with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

> **Note**
>
> Forward access sessions (FAS) in Redshift are valid for 12 hours only. After this period, any connection session using FAS to integrate with other services must be re-established.

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should

not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account. For more information, see Cross-service confused deputy prevention in the *IAM User Guide*.

We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in resource-based policies to limit the permissions that Amazon Redshift gives another service to the resource. If the `aws:SourceArn` value doesn't contain the account ID, such as an Amazon S3 bucket ARN, you must use both global condition context keys to limit permissions. If you use both global condition context keys and the `aws:SourceArn` value contains the account ID, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement.

The following example shows a policy that you can apply to limit the confused deputy problem for Amazon Redshift.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionForRedshift",
    "Effect": "Allow",
    "Principal": {
      "Service": "redshift.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:redshift:us-east-1:123456789012:cluster:my-cluster"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

The following example shows a policy for Amazon Redshift Serverless.

```
{
  "Version": "2012-10-17",
  "Statement": {
```

```
    "Sid": "ConfusedDeputyPreventionForRedshiftServerless",
    "Effect": "Allow",
    "Principal": {
      "Service": "redshift-serverless.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:redshift-serverless:us-east-1:123456789012:workgroup/
my-workgroup"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

Access control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access Amazon Redshift resources. For example, you must have permissions to create an Amazon Redshift cluster, create a snapshot, add an event subscription, and so on.

The following sections describe how to manage permissions for Amazon Redshift. We recommend that you read the overview first.

- Overview of managing access permissions to your Amazon Redshift resources
- Using identity-based policies (IAM policies) for Amazon Redshift

Overview of managing access permissions to your Amazon Redshift resources

Every AWS resource is owned by an AWS account, and permissions to create or access the resources are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles), and some services (such as AWS Lambda) also support attaching permissions policies to resources.

> **Note**
>
> An *account administrator* (or administrator user) is a user with administrator privileges. For more information, see IAM best practices in the *IAM User Guide*.

When granting permissions, you decide who is getting the permissions, which resources they get permissions for, and the specific actions that you want to allow on those resources.

Amazon Redshift resources and operations

Amazon Redshift provides service-specific resources, actions, and condition context keys for use in IAM permission policies.

Amazon Redshift, Amazon Redshift Serverless, Amazon Redshift Data API, and Amazon Redshift query editor v2 access permissions

When you set up Access control, you write permission policies that you can attach to an IAM identity (identity-based policies). For detailed reference information, see the following topics in the *Service Authorization Reference*:

- For Amazon Redshift, see Actions, resources, and condition keys for Amazon Redshift that use the `redshift:` prefix.
- For Amazon Redshift Serverless, see Actions, resources, and condition keys for Amazon Redshift Serverless that use the `redshift-serverless:` prefix.
- For Amazon Redshift Data API, see Actions, resources, and condition keys for Amazon Redshift Data API that use the `redshift-data:` prefix.
- For Amazon Redshift query editor v2, see Actions, resources, and condition keys for AWS SQL Workbench (Amazon Redshift query editor v2) that use the `sqlworkbench:` prefix.

  The query editor v2 includes permission-only actions that don't directly correspond to an API operation. These actions are indicated in the *Service Authorization Reference* with `[permission only]`.

The *Service Authorization Reference* contains information about which API operations can be used in an IAM policy. It also includes the AWS resource for which you can grant the permissions, and condition keys that you can include for fine-grained access control. For more information about conditions, see Using IAM policy conditions for fine-grained access control.

You specify the actions in the policy's `Action` field, the resource value in the policy's `Resource` field, and conditions in the policy's `Condition` field. To specify an action for Amazon Redshift, use the `redshift:` prefix followed by the API operation name (for example, `redshift:CreateCluster`).

Understanding resource ownership

A *resource owner* is the AWS account that created a resource. That is, the resource owner is the AWS account of the *principal entity* (the root account, an IAM user, or an IAM role) that authenticates the request that creates the resource. The following examples illustrate how this works:

- If you use the root account credentials of your AWS account to create a DB cluster, your AWS account is the owner of the Amazon Redshift resource.
- If you create an IAM role in your AWS account with permissions to create Amazon Redshift resources, anyone who can assume the role can create Amazon Redshift resources. Your AWS account, to which the role belongs, owns the Amazon Redshift resources.
- If you create an IAM user in your AWS account and grant permissions to create Amazon Redshift resources to that user, the user can create Amazon Redshift resources. However, your AWS account, to which the user belongs, owns the Amazon Redshift resources. In most cases this method isn't recommended. We recommend creating an IAM role and attaching permissions to the role, then assigning the role to a user.

Managing access to resources

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

> **Note**
>
> This section discusses using IAM in the context of Amazon Redshift. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see What is IAM? in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see AWS IAM policy reference in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as *identity-based* policies (IAM policies) and policies attached to a resource are referred to as *resource-based* policies. Amazon Redshift supports only identity-based policies (IAM policies).

Identity-based policies (IAM policies)

You can assign permissions by attaching polices to an IAM role and then assigning that role to a user or group. The following is an example policy that containing permissions to create, delete, modify, and reboot Amazon Redshift clusters for your AWS account.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid":"AllowManageClusters",
      "Effect":"Allow",
      "Action": [
        "redshift:CreateCluster",
        "redshift:DeleteCluster",
        "redshift:ModifyCluster",
        "redshift:RebootCluster"
      ],
      "Resource":"*"
    }
  ]
}
```

For more information about using identity-based policies with Amazon Redshift, see Using identity-based policies (IAM policies) for Amazon Redshift. For more information about users, groups, roles, and permissions, see Identities (users, groups, and roles) in the *IAM User Guide*.

Resource-based policies

Other services, such as Amazon S3, also support resource-based permissions policies. For example, you can attach a policy to an S3 bucket to manage access permissions to that bucket. Amazon Redshift doesn't support resource-based policies.

Specifying policy elements: Actions, effects, resources, and principals

For each Amazon Redshift resource (see Amazon Redshift resources and operations), the service defines a set of API operations (see Actions). To grant permissions for these API operations,

Amazon Redshift defines a set of actions that you can specify in a policy. Performing an API operation can require permissions for more than one action.

The following are the basic policy elements:

- **Resource** – In a policy, you use an Amazon Resource Name (ARN) to identify the resource to which the policy applies. For more information, see Amazon Redshift resources and operations.
- **Action** – You use action keywords to identify resource operations that you want to allow or deny. For example, the `redshift:DescribeClusters` permission allows the user permissions to perform the Amazon Redshift `DescribeClusters` operation.
- **Effect** – You specify the effect when the user requests the specific action—this can be either allow or deny. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only). Amazon Redshift doesn't support resource-based policies.

To learn more about IAM policy syntax and descriptions, see AWS IAM policy reference in the *IAM User Guide*.

For a table showing all of the Amazon Redshift API actions and the resources that they apply to, see Amazon Redshift, Amazon Redshift Serverless, Amazon Redshift Data API, and Amazon Redshift query editor v2 access permissions.

Specifying conditions in a policy

When you grant permissions, you can use the access policy language to specify the conditions when a policy should take effect. For example, you might want a policy to be applied only after a specific date. For more information about specifying conditions in an access policy language, see IAM JSON policy elements: Condition in the *IAM User Guide*.

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. For details, see Using IAM policy

conditions for fine-grained access control. For a list showing all of condition key values and the Amazon Redshift actions and resources that they apply to, see Amazon Redshift, Amazon Redshift Serverless, Amazon Redshift Data API, and Amazon Redshift query editor v2 access permissions.

Using IAM policy conditions for fine-grained access control

In Amazon Redshift, you can use condition keys to restrict access to resources based on the tags for those resources. The following are common Amazon Redshift condition keys.

Condition key	Description
`aws:RequestTag`	Requires users to include a tag key (name) and value whenever they create a resource. For more information, see aws:RequestTag in the *IAM User Guide*.
`aws:ResourceTag`	Restricts user access to resources based on specific tag keys and values. For more information, see aws:ResourceTag in the *IAM User Guide*.
`aws:TagKeys`	Use this key to compare the tag keys in a request with the keys that you specify in the policy. For more information, see aws:TagKeys in the *IAM User Guide*.

For information on tags, see Tag resources in Amazon Redshift.

For a list of the API actions that support the `redshift:RequestTag` and `redshift:ResourceTag` condition keys, see Amazon Redshift, Amazon Redshift Serverless, Amazon Redshift Data API, and Amazon Redshift query editor v2 access permissions.

The following condition keys can be used with the Amazon Redshift GetClusterCredentials action.

Condition key	Description
`redshift:DurationSeconds`	Limits the number of seconds that can be specified for duration.
`redshift:DbName`	Restricts database names that can be specified.
`redshift:DbUser`	Restricts database user names that can be specified.

Example 1: Restricting access by using the aws:ResourceTag condition key

Use the following IAM policy to let a user modify an Amazon Redshift cluster only for a specific AWS account in the `us-west-2` region with a tag named `environment` with a tag value of `test`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Sid":"AllowModifyTestCluster",
        "Effect": "Allow",
        "Action": "redshift:ModifyCluster",
        "Resource": "arn:aws:redshift:us-west-2:123456789012:cluster:*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/environment": "test"
            }
        }
    }
}
```

Example 2: Restricting access by using the aws:RequestTag condition key

Use the following IAM policy to let a user create an Amazon Redshift cluster only if the command to create the cluster includes a tag named `usage` and a tag value of `production`. The condition with `aws:TagKeys` and the `ForAllValues` modifier specifies that only the keys `costcenter` and `usage` can be specified in the request.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Sid":"AllowCreateProductionCluster",
        "Effect": "Allow",
        "Action": [
            "redshift:CreateCluster",
            "redshift:CreateTags"
        ],
```

```
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/usage": "production"
            },
            "ForAllValues:StringEquals": {
                "aws:TagKeys": [
                    "costcenter",
                    "usage"
                ]
            }
        }
    }
}
```

Using identity-based policies (IAM policies) for Amazon Redshift

This topic provides examples of identity-based policies in which an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles).

> **Important**
>
> We recommend that you first review the introductory topics that explain the basic concepts and options available for you to manage access to your Amazon Redshift resources. For more information, see Overview of managing access permissions to your Amazon Redshift resources.

The following shows an example of a permissions policy. The policy allows a user to create, delete, modify, and reboot all clusters, and then denies permission to delete or modify any clusters where the cluster identifier starts with `production` in AWS Region `us-west-2` and AWS account `123456789012`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Sid":"AllowClusterManagement",
      "Action": [
        "redshift:CreateCluster",
        "redshift:DeleteCluster",
        "redshift:ModifyCluster",
        "redshift:RebootCluster"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Sid":"DenyDeleteModifyProtected",
      "Action": [
        "redshift:DeleteCluster",
        "redshift:ModifyCluster"
      ],
      "Resource": [
        "arn:aws:redshift:us-west-2:123456789012:cluster:production*"
      ],
      "Effect": "Deny"
    }
  ]
}
```

The policy has two statements:

- The first statement grants permissions for a user to a user to create, delete, modify, and reboot clusters. The statement specifies a wildcard character (*) as the `Resource` value so that the policy applies to all Amazon Redshift resources owned by the root AWS account.
- The second statement denies permission to delete or modify a cluster. The statement specifies a cluster Amazon Resource Name (ARN) for the `Resource` value that includes a wildcard character (*). As a result, this statement applies to all Amazon Redshift clusters owned by the root AWS account where the cluster identifier begins with `production`.

AWS managed policies for Amazon Redshift

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. Managed policies grant necessary permissions for common use cases so

you can avoid having to investigate what permissions are needed. For more information, see AWS managed policies in the *IAM User Guide*.

You can also create your own custom IAM policies to allow permissions for Amazon Redshift API operations and resources. You can attach these custom policies to the IAM roles or groups that require those permissions.

The following sections describe AWS managed policies, which you can attach to users in your account, and are specific to Amazon Redshift.

Amazon Redshift updates to AWS managed policies

View details about updates to AWS managed policies for Amazon Redshift since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Amazon Redshift Document history page.

Change	Description	Date
AmazonRedshiftFederatedAuthorization – New policy	Amazon Redshift added a new ease-of-use policy policy for running queries with Amazon Redshift Federated Authorization.	November 21, 2025
AmazonRedshiftServiceLinkedRolePolicy – Update to an existing policy	Permission for the action `lakeformation:GetDataAccess` is added to the managed policy. Adding it grants permission to get federated catalog information from AWS Lake Formation. Additional conditions for the actions `glue:GetCatalog` and `glue:GetCatalogs` are added to the managed policy.	March 13, 2025

Change	Description	Date
AmazonRedshiftServiceLinkedRolePolicy – Update to an existing policy	Permission for the actions `glue:GetCatalog` and `glue:GetCatalogs` are added to the managed policy. Adding them grants permission to get catalog information from AWS Glue.	December 3, 2024
AmazonRedshiftServiceLinkedRolePolicy – Update to an existing policy	Permission for the action `servicequotas:GetServiceQuota` is added to the managed policy. This gives permission to access quotas or limits.	March 8, 2024
AmazonRedshiftQueryEditorV2FullAccess – Update to an existing policy	Permission for the actions `redshift-serverless:ListNamespaces` and `redshift-serverless:ListWorkgroups` are added to the managed policy. Adding them grants permission to list serverless namespaces and serverless workgroups in the Amazon Redshift data warehouse.	February 21, 2024

Change	Description	Date
AmazonRedshiftQueryEditorV2NoSharing – Update to an existing policy	Permission for the actions `redshift-serverless:ListNamespaces` and `redshift-serverless:ListWorkgroups` are added to the managed policy. Adding them grants permission to list serverless namespaces and serverless workgroups in the Amazon Redshift data warehouse.	February 21, 2024
AmazonRedshiftQueryEditorV2ReadSharing – Update to an existing policy	Permission for the actions `redshift-serverless:ListNamespaces` and `redshift-serverless:ListWorkgroups` are added to the managed policy. Adding them grants permission to list serverless namespaces and serverless workgroups in the Amazon Redshift data warehouse.	February 21, 2024
AmazonRedshiftQueryEditorV2ReadWriteSharing – Update to an existing policy	Permission for the actions `redshift-serverless:ListNamespaces` and `redshift-serverless:ListWorkgroups` are added to the managed policy. Adding them grants permission to list serverless namespaces and serverless workgroups in the Amazon Redshift data warehouse.	February 21, 2024

Change	Description	Date
AmazonRedshiftReadOnlyAccess – Update to an existing policy	Permission for the action `redshift:ListRecommendations` is added to the managed policy. This grants permission to list Amazon Redshift Advisor recommendations.	February 7, 2024
AmazonRedshiftServiceLinkedRolePolicy – Update to an existing policy	Permission for the actions `ec2:AssignIpv6Addresses` and `ec2:UnassignIpv6Addresses` are added to the managed policy. Adding them grants permission to assign and unassign IP addresses.	October 31, 2023
AmazonRedshiftQueryEditorV2NoSharing – Update to an existing policy	Permission for the actions `sqlworkbench:GetAutocompletionMetadata` and `sqlworkbench:GetAutocompletionResource` are added to the managed policy. Adding them grants permission to generate and retrieve database information for auto-completion of SQL while editing queries.	August 16, 2023

Change	Description	Date
AmazonRedshiftQueryEditorV2ReadSharing – Update to an existing policy	Permission for the actions `sqlworkbench:GetAutocompletionMetadata` and `sqlworkbench:GetAutocompletionResource` are added to the managed policy. Adding them grants permission to generate and retrieve database information for auto-completion of SQL while editing queries.	August 16, 2023
AmazonRedshiftQueryEditorV2ReadWriteSharing – Update to an existing policy	Permission for the actions `sqlworkbench:GetAutocompletionMetadata` and `sqlworkbench:GetAutocompletionResource` are added to the managed policy. Adding them grants permission to generate and retrieve database information for auto-completion of SQL while editing queries.	August 16, 2023

Change	Description	Date
AmazonRedshiftServiceLinkedRolePolicy – Update to an existing policy	Permissions for actions on AWS Secrets Manager to create and manage secrets are added to the managed policy. Added permissions are the following: • `secretsmanager:GetRandomPassword` • `secretsmanager:DescribeSecret` • `secretsmanager:PutSecretValue` • `secretsmanager:UpdateSecret` • `secretsmanager:UpdateSecretVersionStage` • `secretsmanager:RotateSecret` • `secretsmanager:DeleteSecret`	August 14, 2023

Change	Description	Date
AmazonRedshiftServiceLinkedRolePolicy – Update to an existing policy	Permissions for actions on Amazon EC2 to create and manage security groups and routing rules are removed from the managed policy. These permissions pertained to creating subnets and VPCs. Removed permissions are the following: • `ec2:AuthorizeSecurityGroupEgress` • `ec2:AuthorizeSecurityGroupIngress` • `ec2:UpdateSecurityGroupRuleDescriptionsEgress` • `ec2:ReplaceRouteTableAssociation` • `ec2:CreateRouteTable` • `ec2:AttachInternetGateway` • `ec2:UpdateSecurityGroupRuleDescriptionsIngress` • `ec2:AssociateRouteTable` • `ec2:RevokeSecurityGroupIngress` • `ec2:CreateRoute`	May 08, 2023

Change	Description	Date
	• `ec2:CreateSecurityGroup` • `ec2:RevokeSecurityGroupEgress` • `ec2:ModifyVpcAttribute` • `ec2:CreateSubnet` • `ec2:CreateInternetGateway` • `ec2:CreateVpc` These were associated with the *Purpose:RedshiftMigrateToVpc* resource tag. The tag limited the scope of permissions to tasks for Amazon EC2 Classic to Amazon EC2 VPC migration. For more information about resource tags, see Controlling access to AWS resources using tags.	
AmazonRedshiftDataFullAccess – Update to an existing policy	Permission for the action `redshift:GetClusterCredentialsWithIAM` is added to the managed policy. Adding it grants permission to get enhanced temporary credentials to access an Amazon Redshift database by the specified AWS account.	April 7, 2023

Change	Description	Date
AmazonRedshiftServiceLinkedRolePolicy – Update to an existing policy	Permissions for the actions on Amazon EC2 for creation and management of security group rules are added to the managed policy. These security groups and rules ares specifically associated with the Amazon Redshift `aws:RequestTag/Redshift` resource tag. This limits the scope of the permissions to specific Amazon Redshift resources.	April 06, 2023
AmazonRedshiftQueryEditorV2NoSharing – Update to an existing policy	Permission for the action `sqlworkbench:GetSchemaInference` is added to the managed policy. Adding it grants permission to get the columns and data types inferred from a file.	March 21, 2023
AmazonRedshiftQueryEditorV2ReadSharing – Update to an existing policy	Permission for the action `sqlworkbench:GetSchemaInference` is added to the managed policy. Adding it grants permission to get the columns and data types inferred from a file.	March 21, 2023

Change	Description	Date
AmazonRedshiftQueryEditorV2ReadWriteSharing – Update to an existing policy	Permission for the action `sqlworkbench:GetSchemaInference` is added to the managed policy. Adding it grants permission to get the columns and data types inferred from a file.	March 21, 2023
AmazonRedshiftQueryEditorV2NoSharing – Update to an existing policy	Permission for the action `sqlworkbench:AssociateNotebookWithTab` is added to the managed policy. Adding it grants permission to create and update tabs linked to a user's own notebook.	February 2, 2023
AmazonRedshiftQueryEditorV2ReadSharing – Update to an existing policy	Permission for the action `sqlworkbench:AssociateNotebookWithTab` is added to the managed policy. Adding it grants permission to create and update tabs linked to a user's own notebook or to a notebook that is shared with them.	February 2, 2023

Change	Description	Date
AmazonRedshiftQueryEditorV2ReadWriteSharing – Update to an existing policy	Permission for the action `sqlworkbench:AssociateNotebookWithTab` is added to the managed policy. Adding it grants permission to create and update tabs linked to a user's own notebook or to a notebook that is shared with them.	February 2, 2023

Change	Description	Date
AmazonRedshiftQueryEditorV2NoSharing – Update to an existing policy	To grant permission to use notebooks, Amazon Redshift added permission for the following actions: • sqlworkbench:ListNotebooks • sqlworkbench:CreateNotebook • sqlworkbench:DuplicateNotebook • sqlworkbench:CreateNotebookFromVersion • sqlworkbench:ImportNotebook • sqlworkbench:GetNotebook • sqlworkbench:UpdateNotebook • sqlworkbench:DeleteNotebook • sqlworkbench:CreateNotebookCell • sqlworkbench:DeleteNotebookCell • sqlworkbench:UpdateNotebookCellContent • sqlworkbench:UpdateNotebookCellLayout	October 17, 2022

Change	Description	Date
	• sqlworkbench:BatchGetNotebookCell • sqlworkbench:ListNotebookVersions • sqlworkbench:CreateNotebookVersion • sqlworkbench:GetNotebookVersion • sqlworkbench:DeleteNotebookVersion • sqlworkbench:RestoreNotebookVersion • sqlworkbench:ExportNotebook	

Change	Description	Date
AmazonRedshiftQueryEditorV2ReadSharing – Update to an existing policy	To grant permission to use notebooks, Amazon Redshift added permission for the following actions: • `sqlworkbench:ListNotebooks` • `sqlworkbench:CreateNotebook` • `sqlworkbench:DuplicateNotebook` • `sqlworkbench:CreateNotebookFromVersion` • `sqlworkbench:ImportNotebook` • `sqlworkbench:GetNotebook` • `sqlworkbench:UpdateNotebook` • `sqlworkbench:DeleteNotebook` • `sqlworkbench:CreateNotebookCell` • `sqlworkbench:DeleteNotebookCell` • `sqlworkbench:UpdateNotebookCellContent` • `sqlworkbench:UpdateNotebookCellLayout`	October 17, 2022

Change	Description	Date
	• sqlworkbench:BatchGetNotebookCell • sqlworkbench:ListNotebookVersions • sqlworkbench:CreateNotebookVersion • sqlworkbench:GetNotebookVersion • sqlworkbench:DeleteNotebookVersion • sqlworkbench:RestoreNotebookVersion • sqlworkbench:ExportNotebook	

Change	Description	Date
AmazonRedshiftQueryEditorV2ReadWriteSharing – Update to an existing policy	To grant permission to use notebooks, Amazon Redshift added permission for the following actions: • sqlworkbench:ListNotebooks • sqlworkbench:CreateNotebook • sqlworkbench:DuplicateNotebook • sqlworkbench:CreateNotebookFromVersion • sqlworkbench:ImportNotebook • sqlworkbench:GetNotebook • sqlworkbench:UpdateNotebook • sqlworkbench:DeleteNotebook • sqlworkbench:CreateNotebookCell • sqlworkbench:DeleteNotebookCell • sqlworkbench:UpdateNotebookCellContent • sqlworkbench:UpdateNotebookCellLayout	October 17, 2022

Change	Description	Date
	• `sqlworkbench:BatchGetNotebookCell` • `sqlworkbench:ListNotebookVersions` • `sqlworkbench:CreateNotebookVersion` • `sqlworkbench:GetNotebookVersion` • `sqlworkbench:DeleteNotebookVersion` • `sqlworkbench:RestoreNotebookVersion` • `sqlworkbench:ExportNotebook`	
AmazonRedshiftServiceLinkedRolePolicy – Update to an existing policy	Amazon Redshift added the namespace `AWS/Redshift` to allow publishing metrics to CloudWatch.	September 7, 2022
AmazonRedshiftQueryEditorV2NoSharing – Update to an existing policy	Amazon Redshift added permission to the actions `sqlworkbench:ListQueryExecutionHistory` and `sqlworkbench:GetQueryExecutionHistory`. This grants permission to see query history.	August 30, 2022

Change	Description	Date
AmazonRedshiftQueryEditorV2ReadSharing – Update to an existing policy	Amazon Redshift added permission to the actions `sqlworkbench:ListQueryExecutionHistory` and `sqlworkbench:GetQueryExecutionHistory`. This grants permission to see query history.	August 30, 2022
AmazonRedshiftQueryEditorV2ReadWriteSharing – Update to an existing policy	Amazon Redshift added permission to the actions `sqlworkbench:ListQueryExecutionHistory` and `sqlworkbench:GetQueryExecutionHistory`. This grants permission to see query history.	August 30, 2022
AmazonRedshiftFullAccess – Update to an existing policy	Permissions for Amazon Redshift Serverless are added to the existing AmazonRedshiftFullAccess managed policy.	July 22, 2022

Change	Description	Date
AmazonRedshiftDataFullAccess – Update to an existing policy	Amazon Redshift updated `redshift-serverless:GetCredentials` default scoping condition of tag `aws:ResourceTag/RedshiftDataFullAccess` permission from `StringEquals` to `StringLike` to grant access to resources tagged with tag key `RedshiftDataFullAccess` and any tag value.	July 11, 2022
AmazonRedshiftDataFullAccess – Update to an existing policy	Amazon Redshift added new permissions to allow `redshift-serverless:GetCredentials` for temporary credentials to Amazon Redshift Serverless.	July 8, 2022
AmazonRedshiftQueryEditorV2NoSharing – Update to an existing policy	Amazon Redshift added permission to the action `sqlworkbench:GetAccountSettings`. This grants permission to get account settings.	June 15, 2022
AmazonRedshiftQueryEditorV2ReadSharing – Update to an existing policy	Amazon Redshift added permission to the action `sqlworkbench:GetAccountSettings`. This grants permission to get account settings.	June 15, 2022

Change	Description	Date
AmazonRedshiftQueryEditorV2ReadWriteSharing – Update to an existing policy	Amazon Redshift added permission to the action `sqlworkbench:GetAccountSettings` . This grants permission to get account settings.	June 15, 2022
AmazonRedshiftServiceLinkedRolePolicy – Update to an existing policy	To enable public access to new Amazon Redshift Serverless endpoints, Amazon Redshift allocates and associates Elastic IP addresses to the VPC endpoint's Elastic network interface in the customer account. It does this via permissions provided through the service linked role. To enable this use case, actions to allocate and release an Elastic IP address are added to the Amazon Redshift Serverless service linked role.	May 26, 2022
AmazonRedshiftQueryEditorV2FullAccess – Update to an existing policy	Permissions to the action `sqlworkbench:ListTaggedResources` . It is scoped specifically to Amazon Redshift query editor v2 resources. This policy update gives the right to call `tag:GetResources` only through query editor v2.	February 22, 2022

Change	Description	Date
AmazonRedshiftQueryEditorV2NoSharing – Update to an existing policy	Permissions to the action `sqlworkbench:ListTaggedResources` . It is scoped specifically to Amazon Redshift query editor v2 resources. This policy update gives the right to call `tag:GetResources` only through query editor v2.	February 22, 2022
AmazonRedshiftQueryEditorV2ReadSharing – Update to an existing policy	Permissions to the action `sqlworkbench:ListTaggedResources` . It is scoped specifically to Amazon Redshift query editor v2 resources. This policy update gives the right to call `tag:GetResources` only through query editor v2.	February 22, 2022
AmazonRedshiftQueryEditorV2ReadWriteSharing – Update to an existing policy	Permissions to the action `sqlworkbench:ListTaggedResources` . It is scoped specifically to Amazon Redshift query editor v2 resources. This policy update gives the right to call `tag:GetResources` only through query editor v2.	February 22, 2022

Change	Description	Date
AmazonRedshiftQueryEditorV2ReadSharing – Update to an existing policy	Permission for the action `sqlworkbench:AssociateQueryWithTab` is added to the managed policy. Adding it allows customers to create editor tabs linked to a query that is shared with them.	February 22, 2022
AmazonRedshiftServiceLinkedRolePolicy – Update to an existing policy	Amazon Redshift added permissions for new actions to allow management of Amazon Redshift network and VPC resources.	November 22, 2021
AmazonRedshiftAllCommandsFullAccess – New policy	Amazon Redshift added a new policy to allow using the IAM role created from the Amazon Redshift console and set it as default for the cluster to run the COPY from Amazon S3, UNLOAD, CREATE EXTERNAL SCHEMA, CREATE EXTERNAL FUNCTION, CREATE MODEL, or CREATE LIBRARY commands.	November 18, 2021
AmazonRedshiftServiceLinkedRolePolicy – Update to an existing policy	Amazon Redshift added permissions for new actions to allow management of Amazon Redshift CloudWatch log groups and log streams, including audit-log export.	November 15, 2021

Change	Description	Date
AmazonRedshiftFullAccess – Update to an existing policy	Amazon Redshift added new permissions to allow model explainability, DynamoDB, Redshift Spectrum, and Amazon RDS federation.	October 07, 2021
AmazonRedshiftQueryEditorV2FullAccess – New policy	Amazon Redshift added a new policy to allow full access to Amazon Redshift query editor v2.	September 24, 2021
AmazonRedshiftQueryEditorV2NoSharing – New policy	Amazon Redshift added a new policy to allow using Amazon Redshift query editor v2 without sharing resources.	September 24, 2021
AmazonRedshiftQueryEditorV2ReadSharing – New policy	Amazon Redshift added a new policy to allow read sharing within Amazon Redshift query editor v2.	September 24, 2021
AmazonRedshiftQueryEditorV2ReadWriteSharing – New policy	Amazon Redshift added a new policy to allow read and update sharing within Amazon Redshift query editor v2.	September 24, 2021
AmazonRedshiftFullAccess – Update to an existing policy	Amazon Redshift added new permissions to allow `sagemaker:*Job*` .	August 18, 2021
AmazonRedshiftDataFullAccess – Update to an existing policy	Amazon Redshift added new permissions to allow `AuthorizeDataShare` .	August 12, 2021

Change	Description	Date
AmazonRedshiftDataFullAccess – Update to an existing policy	Amazon Redshift added new permissions to allow `BatchExecuteStatement` .	July 27, 2021
Amazon Redshift started tracking changes	Amazon Redshift started tracking changes for its AWS managed policies.	July 27, 2021

AmazonRedshiftReadOnlyAccess

Grants read-only access to all Amazon Redshift resources for an AWS account.

You can find the AmazonRedshiftReadOnlyAccess policy on the IAM console and AmazonRedshiftReadOnlyAccess in the *AWS Managed Policy Reference Guide*.

AmazonRedshiftFullAccess

Grants full access to all Amazon Redshift resources for an AWS account. Additionally, this policy grants full access to all Amazon Redshift Serverless resources.

You can find the AmazonRedshiftFullAccess policy on the IAM console and AmazonRedshiftFullAccess in the *AWS Managed Policy Reference Guide*.

AmazonRedshiftQueryEditor

Grants full access to the query editor on the Amazon Redshift console.

You can find the AmazonRedshiftQueryEditor policy on the IAM console and AmazonRedshiftQueryEditor in the *AWS Managed Policy Reference Guide*.

AmazonRedshiftDataFullAccess

Grants full access to the Amazon Redshift Data API operations and resources for an AWS account.

You can find the AmazonRedshiftDataFullAccess policy on the IAM console and AmazonRedshiftDataFullAccess in the *AWS Managed Policy Reference Guide*.

AmazonRedshiftQueryEditorV2FullAccess

Grants full access to the Amazon Redshift query editor v2 operations and resources. This policy also grants access to other required services.

You can find the AmazonRedshiftQueryEditorV2FullAccess policy on the IAM console and AmazonRedshiftQueryEditorV2FullAccess in the *AWS Managed Policy Reference Guide*.

AmazonRedshiftQueryEditorV2NoSharing

Grants the ability to work with Amazon Redshift query editor v2 without sharing resources. This policy also grants access to other required services. The principal using this policy can't tag its resources (such as queries) to share them with other principals in the same AWS account.

You can find the AmazonRedshiftQueryEditorV2NoSharing policy on the IAM console and AmazonRedshiftQueryEditorV2NoSharing in the *AWS Managed Policy Reference Guide*.

AmazonRedshiftQueryEditorV2ReadSharing

Grants the ability to work with Amazon Redshift query editor v2 with limited sharing of resources. This policy also grants access to other required services. The principal using this policy can tag its resources (such as queries) to share them with other principals in the same AWS account. The granted principal can read the resources shared with its team but can't update them.

You can find the AmazonRedshiftQueryEditorV2ReadSharing policy on the IAM console and AmazonRedshiftQueryEditorV2ReadSharing in the *AWS Managed Policy Reference Guide*.

AmazonRedshiftQueryEditorV2ReadWriteSharing

Grants the ability to work with Amazon Redshift query editor v2 with sharing of resources. This policy also grants access to other required services. The principal using this policy can tag its resources (such as queries) to share them with other principals in the same AWS account. The granted principal can read and update the resources shared with its team.

You can find the AmazonRedshiftQueryEditorV2ReadWriteSharing policy on the IAM console and AmazonRedshiftQueryEditorV2ReadWriteSharing in the *AWS Managed Policy Reference Guide*.

AmazonRedshiftServiceLinkedRolePolicy

You can't attach AmazonRedshiftServiceLinkedRolePolicy to your IAM entities. This policy is attached to a service-linked role that allows Amazon Redshift to access account resources. For more information, see Using service-linked roles for Amazon Redshift.

You can find the AmazonRedshiftServiceLinkedRolePolicy policy on the IAM console and AmazonRedshiftServiceLinkedRolePolicy in the *AWS Managed Policy Reference Guide*.

AmazonRedshiftAllCommandsFullAccess

Grants the ability to use the IAM role created from the Amazon Redshift console and set it as default for the cluster to run the COPY from Amazon S3, UNLOAD, CREATE EXTERNAL SCHEMA, CREATE EXTERNAL FUNCTION, and CREATE MODEL commands. The policy also grants permissions to run SELECT statements for related services, such as Amazon S3, CloudWatch Logs, Amazon SageMaker AI, or AWS Glue.

You can find the AmazonRedshiftAllCommandsFullAccess policy on the IAM console and AmazonRedshiftAllCommandsFullAccess in the *AWS Managed Policy Reference Guide*.

AmazonRedshiftFederatedAuthorization

The policy consolidates IAM actions needed to run a query against a Glue Data Catalog database with Amazon Redshift Federated Permissions. Such query goes through AWS Glue and therefore needs Get actions on catalog objects to discover the objects, and Create, Update, Rename and Delete actions to modify the objects. Note that the resources are managed by Amazon Redshift, therefore the principal will also need Redshift permissions to complete the query. `glue:FederateAuthorization` action allows AWS Glue to delegate authorization decisions on the catalog objects to Amazon Redshift.

This policy allows the principal to run queries against the catalog with Amazon Redshift Federated Permissions, but does not allow Registering and Unregistering the Amazon Redshift namespace to AWS Glue. Refer to documentation on IAM Policy Requirements for Amazon Redshift Federated Permissions Setup.

You can find the AmazonRedshiftFederatedAuthorization policy on the IAM console and AmazonRedshiftFederatedAuthorization in the *AWS Managed Policy Reference Guide*.

You can also create your own custom IAM policies to allow permissions for Amazon Redshift API operations and resources. You can attach these custom policies to the IAM roles or groups that require those permissions.

Permissions required to use Redshift Spectrum

Amazon Redshift Spectrum requires permissions to other AWS services to access resources. For details about permissions in IAM policies for Redshift Spectrum, see IAM policies for Amazon Redshift Spectrum in the *Amazon Redshift Database Developer Guide.*

Permissions required to use the Amazon Redshift console

For a user to work with the Amazon Redshift console, that user must have a minimum set of permissions that allows the user to describe the Amazon Redshift resources for their AWS account. These permissions must also allow the user to describe other related information, including Amazon EC2 security, Amazon CloudWatch, Amazon SNS, and network information.

If you create an IAM policy that is more restrictive than the minimum required permissions, the console doesn't function as intended for users with that IAM policy. To ensure that those users can still use the Amazon Redshift console, also attach the `AmazonRedshiftReadOnlyAccess` managed policy to the user. How to do this is described in AWS managed policies for Amazon Redshift.

For information to give a user access to the query editor on the Amazon Redshift console, see Permissions required to use the Amazon Redshift console query editor.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the Amazon Redshift API.

Permissions required to use the Amazon Redshift console query editor

For a user to work with the Amazon Redshift query editor, that user must have a minimum set of permissions to Amazon Redshift and Amazon Redshift Data API operations. To connect to a database using a secret, you must also have Secrets Manager permissions.

To give a user access to the query editor on the Amazon Redshift console, attach the `AmazonRedshiftQueryEditor` and `AmazonRedshiftReadOnlyAccess` AWS managed policies. The `AmazonRedshiftQueryEditor` policy allows the user permission to retrieve the results of only their own SQL statements. That is, statements submitted by the same `aws:userid` as shown in this section of the `AmazonRedshiftQueryEditor` AWS managed policy.

```
{
    "Sid":"DataAPIIAMStatementPermissionsRestriction",
```

```
    "Action": [
        "redshift-data:GetStatementResult",
        "redshift-data:CancelStatement",
        "redshift-data:DescribeStatement",
        "redshift-data:ListStatements"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "redshift-data:statement-owner-iam-userid": "${aws:userid}"
        }
    }
}
```

To allow a user to retrieve the results of SQL statements of others in the same IAM role, create your own policy without the condition to limit access to the current user. Also limit access to change a policy to an administrator.

Permissions required to use the query editor v2

For a user to work with the Amazon Redshift query editor v2, that user must have a minimum set of permissions to Amazon Redshift, the query editor v2 operations, and other AWS services such as AWS Key Management Service, AWS Secrets Manager, and tagging service.

To give a user full access to the query editor v2, attach the `AmazonRedshiftQueryEditorV2FullAccess` AWS managed policy. The `AmazonRedshiftQueryEditorV2FullAccess` policy allows the user permission to share query editor v2 resources, such as queries, with others in the same team. For details about how access to query editor v2 resources are controlled, see the definition of the specific managed policy for query editor v2 in the IAM console.

Some Amazon Redshift query editor v2 AWS managed policies use AWS tags within conditions to scope access to resources. Within query editor v2, sharing queries is based on the tag key and value `"aws:ResourceTag/sqlworkbench-team": "${aws:PrincipalTag/sqlworkbench-team}"` in the IAM policy attached to principal (the IAM role). Principals in the same AWS account with the same tag value (for example, `accounting-team`), are on the same team in query editor v2. You can only be associated with one team at a time. A user with administrative permissions can set up teams in the IAM console by giving all team members the same value for the `sqlworkbench-team` tag. If the tag value of the `sqlworkbench-team` is changed for an IAM

user or an IAM role, there might be a delay until the change is reflected in shared resources. If the tag value of a resource (such as a query) is changed, again there might be a delay until the change is reflected. Team members must also have the `tag:GetResources` permission to share.

Example: To add the `accounting-team` tag for an IAM role

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane of the console, choose **Roles** and then choose the name of the role that you want to edit.
3. Choose the **Tags** tab and then choose **Add tags**.
4. Add the tag key **sqlworkbench-team** and the value `accounting-team`.
5. Choose **Save changes**.

   Now when an IAM principal (with this IAM role attached) shares a query with the team, other principals with the same `accounting-team` tag value can view the query.

For more information on how to attach a tag to a principal, including IAM roles and IAM users, see Tagging IAM resources in the *IAM User Guide*.

You can also set up teams at the session level using an Identity Provider (IdP). This allows multiple users using the same IAM role to have different team. The IAM role trust policy must allow the `sts:TagSession` operation. For more information, see Permissions required to add session tags in the *IAM User Guide*. Add the principal tag attribute to the SAML assertion provided by your IdP.

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:sqlworkbench-
team">
    <AttributeValue>accounting-team</AttributeValue>
</Attribute>
```

Follow the instructions for your Identity provider (IdP) to populate the SAML attribute with the content coming from your directory. For more information about Identity providers (IdPs) and Amazon Redshift, see Using IAM authentication to generate database user credentials and Identity providers and federation in the *IAM User Guide*.

The `sqlworkbench:CreateNotebookVersion` grants permission to get the current content of notebook cells and create a notebook version on your account. Meaning, at the time of version creation, the current content of the notebook is the same as the version's content. Later on,

the content of the cells in the version stay the same as the current notebook is updated. The `sqlworkbench:GetNotebookVersion` grants permission to get a version of the notebook. A user who doesn't have `sqlworkbench:BatchGetNotebookCell` permission but has `sqlworkbench:CreateNotebookVersion` and `sqlworkbench:GetNotebookVersion` permissions on a notebook has access to notebook cells in the version. This user without the `sqlworkbench:BatchGetNotebookCell` permission is still able to retrieve the content of a notebook's cells by first creating a version and then getting this created version.

Permissions required to use the Amazon Redshift scheduler

When you use the Amazon Redshift scheduler, you set up an IAM role with a trust relationship to the Amazon Redshift scheduler (**`scheduler.redshift.amazonaws.com`**) to allow the scheduler to assume permissions on your behalf. You also attach a policy (permissions) to the role for the Amazon Redshift API operations that you want to schedule.

The following example shows the policy document in JSON format to set up a trust relationship with the Amazon Redshift scheduler and Amazon Redshift.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "scheduler.redshift.amazonaws.com",
                    "redshift.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

For more information about trust entities, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

You also must add permission for the Amazon Redshift operations you want to schedule.

For the scheduler to use the `ResizeCluster` operation, add a permission that is similar to the following to your IAM policy. Depending on your environment, you might want to make the policy more restrictive.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "redshift:ResizeCluster",
            "Resource": "*"
        }
    ]
}
```

For the steps to create a role for the Amazon Redshift scheduler, see Creating a role for an AWS service (console) in the *IAM User Guide*. Make these choices when you create a role in the IAM console:

- For **Choose the service that will use this role**: Choose **Redshift**.
- For **Select your use case**: Choose **Redshift - Scheduler**.
- Create or attach a policy to the role that allows an Amazon Redshift operation to be scheduled. Choose **Create policy** or modify the role to attach a policy. Enter the JSON policy for the operation that is to be scheduled.
- After you create the role, edit the **Trust Relationship** of the IAM role to include the service `redshift.amazonaws.com`.

The IAM role you create has trusted entities of `scheduler.redshift.amazonaws.com` and `redshift.amazonaws.com`. It also has an attached policy that allows a supported Amazon Redshift API action, such as, `"redshift:ResizeCluster"`.

Permissions required to use the Amazon EventBridge scheduler

When you use the Amazon EventBridge scheduler, you set up an IAM role with a trust relationship to the EventBridge scheduler (**`events.amazonaws.com`**) to allow the scheduler to assume permissions on your behalf. You also attach a policy (permissions) to the role for the Amazon Redshift Data API operations that you want to schedule and a policy for Amazon EventBridge operations.

You use the EventBridge scheduler when you create scheduled queries with the Amazon Redshift query editor on the console.

You can create an IAM role to run scheduled queries on the IAM console. In this IAM role, attach `AmazonEventBridgeFullAccess` and `AmazonRedshiftDataFullAccess`.

The following example shows the policy document in JSON format to set up a trust relationship with the EventBridge scheduler.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "events.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

For more information about trust entities, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

For the steps to create a role for the EventBridge scheduler, see Creating a role for an AWS service (console) in the *IAM User Guide*. Make these choices when you create a role in the IAM console:

- For **Choose the service that will use this role**: Choose **CloudWatch Events**.
- For **Select your use case**: Choose **CloudWatch Events**.
- Attach the following permission policies: `AmazonEventBridgeFullAccess` and `AmazonRedshiftDataFullAccess`.

The IAM role that you create has a trusted entity of `events.amazonaws.com`. It also has an attached policy that allows supported Amazon Redshift Data API actions, such as, `"redshift-data:*"`.

Permissions required to use Amazon Redshift machine learning (ML)

Following, you can find a description of the permissions required to use Amazon Redshift machine learning (ML) for different use cases.

For your users to use Amazon Redshift ML with Amazon SageMaker AI, create an IAM role with a more restrictive policy than the default. You can use the policy following. You can also modify this policy to meet your needs.

The following policy shows the permissions required to run SageMaker AI Autopilot with model explainability from Amazon Redshift.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateTrainingJob",
                "sagemaker:CreateAutoMLJob",
                "sagemaker:CreateCompilationJob",
                "sagemaker:CreateEndpoint",
                "sagemaker:DescribeAutoMLJob",
                "sagemaker:DescribeTrainingJob",
                "sagemaker:DescribeCompilationJob",
                "sagemaker:DescribeProcessingJob",
                "sagemaker:DescribeTransformJob",
                "sagemaker:ListCandidatesForAutoMLJob",
                "sagemaker:StopAutoMLJob",
```

```
                "sagemaker:StopCompilationJob",
                "sagemaker:StopTrainingJob",
                "sagemaker:DescribeEndpoint",
                "sagemaker:InvokeEndpoint",
                "sagemaker:StopProcessingJob",
                "sagemaker:CreateModel",
                "sagemaker:CreateProcessingJob"
            ],
            "Resource": [
                "arn:aws:sagemaker:*:*:model/*redshift*",
                "arn:aws:sagemaker:*:*:training-job/*redshift*",
                "arn:aws:sagemaker:*:*:automl-job/*redshift*",
                "arn:aws:sagemaker:*:*:compilation-job/*redshift*",
                "arn:aws:sagemaker:*:*:processing-job/*redshift*",
                "arn:aws:sagemaker:*:*:transform-job/*redshift*",
                "arn:aws:sagemaker:*:*:endpoint/*redshift*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/sagemaker/Endpoints/*redshift*",
                "arn:aws:logs:*:*:log-group:/aws/sagemaker/ProcessingJobs/
*redshift*",
                "arn:aws:logs:*:*:log-group:/aws/sagemaker/TrainingJobs/
*redshift*",
                "arn:aws:logs:*:*:log-group:/aws/sagemaker/TransformJobs/
*redshift*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
```

```
                    "cloudwatch:namespace": [
                        "SageMaker",
                        "/aws/sagemaker/Endpoints",
                        "/aws/sagemaker/ProcessingJobs",
                        "/aws/sagemaker/TrainingJobs",
                        "/aws/sagemaker/TransformJobs"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetAuthorizationToken",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetBucketAcl",
                "s3:GetBucketCors",
                "s3:GetEncryptionConfiguration",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:ListMultipartUploadParts",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject",
                "s3:PutBucketAcl",
                "s3:PutBucketCors",
                "s3:DeleteObject",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket"
            ],
            "Resource": [
                "arn:aws:s3:::redshift-downloads",
                "arn:aws:s3:::redshift-downloads/*",
                "arn:aws:s3:::*redshift*",
                "arn:aws:s3:::*redshift*/*"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetBucketAcl",
                "s3:GetBucketCors",
                "s3:GetEncryptionConfiguration",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:ListMultipartUploadParts",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject",
                "s3:PutBucketAcl",
                "s3:PutBucketCors",
                "s3:DeleteObject",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {
                    "s3:ExistingObjectTag/Redshift": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "redshift.amazonaws.com",
                        "sagemaker.amazonaws.com"
                    ]
                }
            }
        }
    ]
```

```
}
```

The following policy shows the full minimal permissions to allow access to Amazon DynamoDB, Redshift Spectrum and Amazon RDS federation.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateTrainingJob",
                "sagemaker:CreateAutoMLJob",
                "sagemaker:CreateCompilationJob",
                "sagemaker:CreateEndpoint",
                "sagemaker:DescribeAutoMLJob",
                "sagemaker:DescribeTrainingJob",
                "sagemaker:DescribeCompilationJob",
                "sagemaker:DescribeProcessingJob",
                "sagemaker:DescribeTransformJob",
                "sagemaker:ListCandidatesForAutoMLJob",
                "sagemaker:StopAutoMLJob",
                "sagemaker:StopCompilationJob",
                "sagemaker:StopTrainingJob",
                "sagemaker:DescribeEndpoint",
                "sagemaker:InvokeEndpoint",
                "sagemaker:StopProcessingJob",
                "sagemaker:CreateModel",
                "sagemaker:CreateProcessingJob"
            ],
            "Resource": [
                "arn:aws:sagemaker:*:*:model/*redshift*",
                "arn:aws:sagemaker:*:*:training-job/*redshift*",
                "arn:aws:sagemaker:*:*:automl-job/*redshift*",
                "arn:aws:sagemaker:*:*:compilation-job/*redshift*",
                "arn:aws:sagemaker:*:*:processing-job/*redshift*",
                "arn:aws:sagemaker:*:*:transform-job/*redshift*",
                "arn:aws:sagemaker:*:*:endpoint/*redshift*"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/sagemaker/Endpoints/*redshift*",
                "arn:aws:logs:*:*:log-group:/aws/sagemaker/ProcessingJobs/
*redshift*",
                "arn:aws:logs:*:*:log-group:/aws/sagemaker/TrainingJobs/
*redshift*",
                "arn:aws:logs:*:*:log-group:/aws/sagemaker/TransformJobs/
*redshift*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "SageMaker",
                        "/aws/sagemaker/Endpoints",
                        "/aws/sagemaker/ProcessingJobs",
                        "/aws/sagemaker/TrainingJobs",
                        "/aws/sagemaker/TransformJobs"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetAuthorizationToken",
                "ecr:GetDownloadUrlForLayer"
```

```
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetBucketAcl",
                "s3:GetBucketCors",
                "s3:GetEncryptionConfiguration",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:ListMultipartUploadParts",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject",
                "s3:PutBucketAcl",
                "s3:PutBucketCors",
                "s3:DeleteObject",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket"
            ],
            "Resource": [
                "arn:aws:s3:::redshift-downloads",
                "arn:aws:s3:::redshift-downloads/*",
                "arn:aws:s3:::*redshift*",
                "arn:aws:s3:::*redshift*/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetBucketAcl",
                "s3:GetBucketCors",
                "s3:GetEncryptionConfiguration",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:ListMultipartUploadParts",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject",
                "s3:PutBucketAcl",
                "s3:PutBucketCors",
```

```
                "s3:DeleteObject",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {
                    "s3:ExistingObjectTag/Redshift": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:Scan",
                "dynamodb:DescribeTable",
                "dynamodb:Getitem"
            ],
            "Resource": [
                "arn:aws:dynamodb:*:*:table/*redshift*",
                "arn:aws:dynamodb:*:*:table/*redshift*/index/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ListInstances"
            ],
            "Resource": [
                "arn:aws:elasticmapreduce:*:*:cluster/*redshift*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ListInstances"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {
                    "elasticmapreduce:ResourceTag/Redshift": "true"
                }
            }
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:*:*:function:*redshift*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:DeleteDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:CreateTable",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition"
            ],
            "Resource": [
                "arn:aws:glue:*:*:table/*redshift*/*",
                "arn:aws:glue:*:*:catalog",
                "arn:aws:glue:*:*:database/*redshift*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds"
            ],
```

```
            "Resource": [
                "arn:aws:secretsmanager:*:*:secret:*redshift*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword",
                "secretsmanager:ListSecrets"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:ResourceTag/Redshift": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "redshift.amazonaws.com",
                        "glue.amazonaws.com",
                        "sagemaker.amazonaws.com",
                        "athena.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Optionally, to use a AWS KMS key for encryption, add the following permissions to the policy.

```
{
    "Effect": "Allow",
    "Action": [
```

```
        "kms:CreateGrant",
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:Encrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": [
        "arn:aws:kms:<your-region>:<your-account-id>:key/<your-kms-key>"
    ]
}
```

To allow Amazon Redshift and SageMaker AI to assume the preceding IAM role to interact with other services, add the following trust policy to the role.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "redshift.amazonaws.com",
          "sagemaker.amazonaws.com",
          "forecast.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

In the preceding, the Amazon S3 bucket `redshift-downloads/redshift-ml/` is the location where the sample data used for other steps and examples is stored. You can remove this bucket if you don't need to load data from Amazon S3. Or replace it with other Amazon S3 buckets that you use to load data into Amazon Redshift.

The **`your-account-id`**, **`your-role`**, and **`your-s3-bucket`** values are the account ID, role, and bucket that you specify in your CREATE MODEL command.

Optionally, you can use the AWS KMS keys section of the sample policy if you specify an AWS KMS key for use with Amazon Redshift ML. The **`your-kms-key`** value is the key that you use as part of your CREATE MODEL command.

When you specify a private virtual private cloud (VPC) for a hyperparameter tuning job, add the following permissions.

```
{
            "Effect": "Allow",
            "Action": [
            "ec2:CreateNetworkInterface",
            "ec2:CreateNetworkInterfacePermission",
            "ec2:DeleteNetworkInterface",
            "ec2:DeleteNetworkInterfacePermission",
            "ec2:DescribeNetworkInterfaces",
            "ec2:DescribeVpcs",
            "ec2:DescribeDhcpOptions",
            "ec2:DescribeSubnets",
            "ec2:DescribeSecurityGroups"
            ]
}
```

To work with model explanation, make sure that you have the permissions to call SageMaker AI API operations. We recommend that you use the `AmazonSageMakerFullAccess` managed policy. If you want to create an IAM role with a more restrictive policy, use the policy following.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateEndpoint",
        "sagemaker:CreateEndpointConfig",
        "sagemaker:DeleteEndpoint",
        "sagemaker:DeleteEndpointConfig",
        "sagemaker:DescribeEndpoint",
        "sagemaker:DescribeEndpointConfig",
        "sagemaker:DescribeModel",
```

```
        "sagemaker:InvokeEndpoint",
        "sagemaker:ListTags"
      ],
      "Resource": "*"
    }
  ]
}
```

For more information about the `AmazonSageMakerFullAccess` managed policy, see AmazonSageMakerFullAccess in the *Amazon SageMaker AI Developer Guide*.

If you want to create Forecast models, we recommend that you use the `AmazonForecastFullAccess` managed policy. If you want to use a more restrictive policy, add the following policy to your IAM role.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "forecast:CreateAutoPredictor",
                "forecast:CreateDataset",
                "forecast:CreateDatasetGroup",
                "forecast:CreateDatasetImportJob",
                "forecast:CreateForecast",
                "forecast:CreateForecastExportJob",
                "forecast:DeleteResourceTree",
                "forecast:DescribeAutoPredictor",
                "forecast:DescribeDataset",
                "forecast:DescribeDatasetGroup",
                "forecast:DescribeDatasetImportJob",
                "forecast:DescribeForecast",
                "forecast:DescribeForecastExportJob",
                "forecast:StopResource",
                "forecast:TagResource",
                "forecast:UpdateDatasetGroup"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

If you want to create Amazon Bedrock models, we recommend that you use the `AmazonBedrockFullAccess` managed policy. If you want to use a more restrictive policy, add the following policy to your IAM role.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "bedrock:InvokeModel",
            "Resource": [
                "*",
                "arn:aws:bedrock:us-east-1::foundation-model/*"
            ]
        }
    ]
}
```

For more information about Amazon Redshift ML, see Using machine learning in Amazon Redshift, CREATE MODEL, or CREATE EXTERNAL MODEL.

Permissions for streaming ingestion

Streaming ingestion works with two services. These are Kinesis Data Streams and Amazon MSK.

Permissions required to use streaming ingestion with Kinesis Data Streams

A procedure with a managed-policy example is available at Getting started with streaming ingestion from Amazon Kinesis Data Streams.

Permissions required to use streaming ingestion with Amazon MSK

A procedure with a managed-policy example is available at Getting started with streaming ingestion from Amazon Managed Streaming for Apache Kafka.

Permissions required to use the data sharing API operations

To control access to the data sharing API operations, use IAM action-based policies. For information about how to manage IAM policies, see Managing IAM policies in the *IAM User Guide*.

In particular, suppose that a producer cluster administrator needs to use the `AuthorizeDataShare` call to authorize egress for a datashare outside of an AWS account. In this case, you set up an IAM action-based policy to grant this permission. Use the `DeauthorizeDataShare` call to revoke egress.

When using IAM action-based policies, you can also specify an IAM resource in the policy, such as `DataShareARN`. The following shows the format and an example for `DataShareARN`.

```
arn:aws:redshift:region:account-id:datashare:namespace-guid/datashare-name
arn:aws:redshift:us-east-1:555555555555:datashare:86b5169f-01dc-4a6f-9fbb-e2e24359e9a8/
SalesShare
```

You can restrict `AuthorizeDataShare` access to a specific datashare by specifying the datashare name in the IAM policy.

```
{
  "Statement": [
    {
      "Action": [
        "redshift:AuthorizeDataShare",
      ],
      "Resource": [
        "arn:aws:redshift:us-east-1:555555555555:datashare:86b5169f-01dc-4a6f-9fbb-
e2e24359e9a8/SalesShare"
      ],
      "Effect": "Deny"
    }
  ]
}
```

You can also restrict the IAM policy to all datashares owned by a specific producer cluster. To do this, replace the **`datashare-name`** value in the policy with a wildcard or an asterisk. Keep the cluster's `namespace-guid` value.

```
arn:aws:redshift:us-east-1:555555555555:datashare:86b5169f-01dc-4a6f-9fbb-e2e24359e9a8/
*
```

Following is an IAM policy that prevents an entity from calling `AuthorizeDataShare` on the datashares owned by a specific producer cluster.

```
{
  "Statement": [
    {
      "Action": [
        "redshift:AuthorizeDataShare",
      ],
      "Resource": [
        "arn:aws:redshift:us-east-1:555555555555:datashare:86b5169f-01dc-4a6f-9fbb-
e2e24359e9a8/*"
      ],
      "Effect": "Deny"
    }
  ]
}
```

`DataShareARN` restricts the access based on both the datashare name and the globally unique ID (GUID) for the owning cluster's namespace. It does this by specifying the name as an asterisk.

Resource policies for GetClusterCredentials

To connect to a cluster database using a JDBC or ODBC connection with IAM database credentials, or to programmatically call the `GetClusterCredentials` action, you need permission to call the `redshift:GetClusterCredentials` action with access to a `dbuser` resource.

If you use a JDBC or ODBC connection, instead of `server` and `port` you can specify `cluster_id` and `region`, but to do so your policy must permit the `redshift:DescribeClusters` action with access to the `cluster` resource.

If you call `GetClusterCredentials` with the optional parameters `Autocreate`, `DbGroups`, and `DbName`, make sure to also allow the actions and permit access to the resources listed in the following table.

GetClusterCredentials parameter	Action	Resource
Autocreate	redshift:CreateClusterUser	dbuser
DbGroups	redshift:JoinGroup	dbgroup
DbName	NA	dbname

For more information about resources, see Amazon Redshift resources and operations.

You can also include the following conditions in your policy:

- `redshift:DurationSeconds`
- `redshift:DbName`
- `redshift:DbUser`

Important

For SAML SSO integrations, you may be required to specify an IAM Policy using the `${redshift:DbUser}` variable. In those cases, we strongly recommend the use of a condition statement that ensures a caller cannot obtain credentials for a user which does not match their AWS userid. E.g. `"StringEquals": {"aws:userid":"AIDIODR4TAW7CSEXAMPLE:${redshift:DbUser}"}`. See Example 8: IAM policy for using GetClusterCredentials. For more information about conditions, see Specifying conditions in a policy

Customer managed policy examples

In this section, you can find example user policies that grant permissions for various Amazon Redshift actions. These policies work when you are using the Amazon Redshift API, AWS SDKs, or the AWS CLI.

> **Note**
>
> All examples use the US West (Oregon) Region (`us-west-2`) and contain fictitious account IDs.

Example 1: Allow user full access to all Amazon Redshift actions and resources

The following policy allows access to all Amazon Redshift actions on all resources.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid":"AllowRedshift",
      "Action": [
        "redshift:*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

The value `redshift:*` in the `Action` element indicates all of the actions in Amazon Redshift.

Example 2: Deny a user access to a set of Amazon Redshift actions

By default, all permissions are denied. However, sometimes you need to explicitly deny access to a specific action or set of actions. The following policy allows access to all the Amazon Redshift actions and explicitly denies access to any Amazon Redshift action where the name starts with `Delete`. This policy applies to all Amazon Redshift resources in `us-west-2`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid":"AllowUSWest2Region",
      "Action": [
        "redshift:*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:redshift:us-east-1:111122223333:*"
    },
   {
     "Sid":"DenyDeleteUSWest2Region",
     "Action": [
        "redshift:Delete*"
      ],
      "Effect": "Deny",
      "Resource": "arn:aws:redshift:us-east-1:111122223333:*"
   }
  ]
}
```

Example 3: Allow a user to manage clusters

The following policy allows a user to create, delete, modify, and reboot all clusters, and then denies permission to delete any clusters where the cluster name starts with `protected`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid":"AllowClusterManagement",
      "Action": [
        "redshift:CreateCluster",
        "redshift:DeleteCluster",
        "redshift:ModifyCluster",
```

```
        "redshift:RebootCluster"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Sid":"DenyDeleteProtected",
      "Action": [
        "redshift:DeleteCluster"
      ],
      "Resource": [
        "arn:aws:redshift:us-west-2:123456789012:cluster:protected*"
      ],
      "Effect": "Deny"
    }
  ]
}
```

Example 4: Allow a user to authorize and revoke snapshot access

The following policy allows a user, for example User A, to do the following:

- Authorize access to any snapshot created from a cluster named `shared`.
- Revoke snapshot access for any snapshot created from the `shared` cluster where the snapshot name starts with `revokable`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid":"AllowSharedSnapshots",
      "Action": [
        "redshift:AuthorizeSnapshotAccess"
      ],
      "Resource": [
        "arn:aws:redshift:us-west-2:123456789012:shared/*"
```

```
      ],
      "Effect": "Allow"
    },
    {
      "Sid":"AllowRevokableSnapshot",
      "Action": [
        "redshift:RevokeSnapshotAccess"
      ],
      "Resource": [
        "arn:aws:redshift:us-west-2:123456789012:snapshot:*/revokable*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

If User A has allowed User B to access a snapshot, User B must have a policy such as the following to allow User B to restore a cluster from the snapshot. The following policy allows User B to describe and restore from snapshots, and to create clusters. The name of these clusters must start with `from-other-account`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid":"AllowDescribeSnapshots",
      "Action": [
        "redshift:DescribeClusterSnapshots"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Sid":"AllowUserRestoreFromSnapshot",
      "Action": [
        "redshift:RestoreFromClusterSnapshot"
      ],
```

```
      "Resource": [
        "arn:aws:redshift:us-west-2:123456789012:snapshot:*/*",
        "arn:aws:redshift:us-west-2:444455556666:cluster:from-other-account*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Example 5: Allow a user to copy a cluster snapshot and restore a cluster from a snapshot

The following policy allows a user to copy any snapshot created from the cluster named `big-cluster-1`, and restore any snapshot where the snapshot name starts with `snapshot-for-restore`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid":"AllowCopyClusterSnapshot",
      "Action": [
        "redshift:CopyClusterSnapshot"
      ],
      "Resource": [
        "arn:aws:redshift:us-west-2:123456789012:snapshot:big-cluster-1/*"
      ],
      "Effect": "Allow"
    },
    {
      "Sid":"AllowRestoreFromClusterSnapshot",
      "Action": [
        "redshift:RestoreFromClusterSnapshot"
      ],
      "Resource": [
        "arn:aws:redshift:us-west-2:123456789012:snapshot:*/snapshot-for-
restore*",
        "arn:aws:redshift:us-west-2:123456789012:cluster:*"
      ],
      "Effect": "Allow"
```

```
    }
  ]
}
```

Example 6: Allow a user access to Amazon Redshift, and common actions and resources for related AWS services

The following example policy allows access to all actions and resources for Amazon Redshift, Amazon Simple Notification Service (Amazon SNS), and Amazon CloudWatch. It also allows specified actions on all related Amazon EC2 resources under the account.

> **Note**
>
> Resource-level permissions are not supported for the Amazon EC2 actions that are specified in this example policy.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid":"AllowRedshift",
      "Effect": "Allow",
      "Action": [
        "redshift:*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid":"AllowSNS",
      "Effect": "Allow",
        "Action": [
          "sns:*"
        ],
        "Resource": [
          "*"
```

```
        ]
      },
    {
      "Sid":"AllowCloudWatch",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid":"AllowEC2Actions",
      "Effect": "Allow",
      "Action": [
        "ec2:AllocateAddress",
        "ec2:AssociateAddress",
        "ec2:AttachNetworkInterface",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAddresses",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeInternetGateways",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Example 7: Allow a user to tag resources with the Amazon Redshift console

The following example policy allows a user to tag resources with the Amazon Redshift console using the AWS Resource Groups. This policy can be attached to a user role that invokes the new or original Amazon Redshift console. For more information about tagging, see Tag resources in Amazon Redshift.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid":"TaggingPermissions",
            "Effect": "Allow",
            "Action": [
                "redshift:DeleteTags",
                "redshift:CreateTags",
                "redshift:DescribeTags",
                "tag:UntagResources",
                "tag:TagResources"
            ],
            "Resource": "*"
        }
    ]
}
```

Example 8: IAM policy for using GetClusterCredentials

The following policy uses these sample parameter values:

- Region: `us-west-2`
- AWS Account: `123456789012`
- Cluster name: `examplecluster`

The following policy enables the `GetCredentials`, `CreateClusterUser`, and `JoinGroup` actions. The policy uses condition keys to allow the `GetClusterCredentials` and `CreateClusterUser` actions only when the AWS user ID matches `"AIDIODR4TAW7CSEXAMPLE:${redshift:DbUser}@yourdomain.com"`. IAM access is requested for the `"testdb"` database only. The policy also allows users to join a group named `"common_group"`.

JSON

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

```
  "Statement": [
    {
     "Sid":"GetClusterCredsStatement",
      "Effect": "Allow",
      "Action": [
        "redshift:GetClusterCredentials"
      ],
      "Resource": [
        "arn:aws:redshift:us-west-2:123456789012:dbuser:examplecluster/
${redshift:DbUser}",
        "arn:aws:redshift:us-west-2:123456789012:dbname:examplecluster/testdb",
        "arn:aws:redshift:us-west-2:123456789012:dbgroup:examplecluster/
common_group"
      ],
        "Condition": {
           "StringEquals": {
           "aws:userid":"AIDIODR4TAW7CSEXAMPLE:${redshift:DbUser}@yourdomain.com"
           }
        }
    },
    {
      "Sid":"CreateClusterUserStatement",
      "Effect": "Allow",
      "Action": [
        "redshift:CreateClusterUser"
      ],
      "Resource": [
        "arn:aws:redshift:us-west-2:123456789012:dbuser:examplecluster/
${redshift:DbUser}"
      ],
      "Condition": {
        "StringEquals": {
          "aws:userid":"AIDIODR4TAW7CSEXAMPLE:${redshift:DbUser}@yourdomain.com"
        }
      }
    },
    {
      "Sid":"RedshiftJoinGroupStatement",
      "Effect": "Allow",
      "Action": [
        "redshift:JoinGroup"
      ],
      "Resource": [
```

```
        "arn:aws:redshift:us-west-2:123456789012:dbgroup:examplecluster/
common_group"
      ]
    }
  ]
}
```

The following example shows a policy that allows the IAM role to call the `GetClusterCredentials` operation. Specifying the Amazon Redshift `dbuser` resource grants the role access to the database user name `temp_creds_user` on the cluster named `examplecluster`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "redshift:GetClusterCredentials",
    "Resource": "arn:aws:redshift:us-west-2:123456789012:dbuser:examplecluster/
temp_creds_user"
  }
}
```

You can use a wildcard (*) to replace all, or a portion of, the cluster name, user name, and database group names. The following example allows any user name beginning with `temp_` with any cluster in the specified account.

Important

The statement in the following example specifies a wildcard character (*) as part of the value for the resource so that the policy permits any resource that begins with the specified characters. Using a wildcard character in your IAM policies might be overly permissive. As a best practice, we recommend using the most restrictive policy feasible for your business application.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "redshift:GetClusterCredentials",
    "Resource": "arn:aws:redshift:us-west-2:123456789012:dbuser:*/temp_*"
  }
}
```

The following example shows a policy that allows the IAM role to call the `GetClusterCredentials` operation with the option to automatically create a new user and specify groups the user joins at login. The `"Resource": "*"` clause grants the role access to any resource, including clusters, database users, or user groups.

> **Important**
>
> The statement in the following example specifies a wildcard character (*) as the resource for the given actions, so that the policy permits access to any cluster and database users, and allows creating any user. Using a wildcard character in your IAM policies might be overly permissive. As a best practice, we recommend using the most restrictive policy feasible for your business application.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
             "redshift:GetClusterCredentials",
             "redshift:CreateClusterUser",
  "redshift:JoinGroup"
           ],
    "Resource": "*"
  }
```

```
}
```

For more information, see Amazon Redshift ARN syntax.

Native identity provider (IdP) federation for Amazon Redshift

Managing identities and permissions for Amazon Redshift is made easier with native identity provider federation because it leverages your existing identity provider to simplify authentication and managing permissions. It does this by making it possible to share identity metadata to Redshift from your identity provider. For the first iteration of this feature, the supported identity provider is Microsoft Azure Active Directory (Azure AD).

To configure Amazon Redshift so it can authenticate identities from the third-party identity provider, you register the identity provider with Amazon Redshift. Doing this enables Redshift to authenticate users and roles defined by the identity provider. Thus you can avoid having to perform granular identity management in both your third-party identity provider and in Amazon Redshift, because identity information is shared.

For information about using session roles that are transferred from identity provider (IdP) groups, see PG_GET_SESSION_ROLES in the *Amazon Redshift Database Developer Guide*.

Native identity provider (IdP) federation

To complete the preliminary setup between the identity provider and Amazon Redshift, you perform a couple of steps: First, you register Amazon Redshift as a third-party application with your identity provider, requesting the necessary API permissions. Then you create users and groups in the identity provider. Last, you register the identity provider with Amazon Redshift, using SQL statements, which set authentication parameters that are unique to the identity provider. As part of registering the identity provider with Redshift, you assign a namespace to make sure users and roles are grouped correctly.

With the identity provider registered with Amazon Redshift, communication is set up between Redshift and the identity provider. A client can then pass tokens and authenticate to Redshift as an identity provider entity. Amazon Redshift uses the IdP group membership information to map to Redshift roles. If the user doesn't previously exist in Redshift, the user is created. Roles are created that map to identity provider groups, if they don't exist. The Amazon Redshift administrator grants permission on the roles, and users can run queries and perform other database tasks.

The following steps outline how native identity provider federation works, when a user logs in:

1. When a user logs in using the native IdP option, from the client, the identity provider token is sent from the client to the driver.
2. The user is authenticated. If the user doesn't already exist in Amazon Redshift, a new user is created. Redshift maps the user's identity provider groups to Redshift roles.
3. Permissions are assigned, based on the user's Redshift roles. These are granted to users and roles by an administrator.
4. The user can query Redshift.

Desktop client tools

For instructions on how to use native identity provider federation to connect to Amazon Redshift with Power BI, see the blog post Integrate Amazon Redshift native IdP federation with Microsoft Azure Active Directory (AD) and Power BI. It describes a step-by-step implementation of the Amazon Redshift native IdP setup with Azure AD. It details the steps to set up the client connection for either Power BI Desktop or the Power BI service. The steps include application registration, configuring permissions, and configuring credentials.

To learn how to integrate Amazon Redshift native IdP federation with Azure AD, using Power BI Desktop and JDBC Client-SQL Workbench/J, watch the following video:

For instructions on how to use native identity provider federation to connect to Amazon Redshift with a SQL client, specifically DBeaver or SQL Workbench/J, see the blog post Integrate Amazon Redshift native IdP federation with Microsoft Azure AD using a SQL client.

Limitations

These limitations apply:

- Amazon Redshift drivers support `BrowserIdcAuthPlugin` starting from the following versions:
  - Amazon Redshift JDBC driver v2.1.0.30
  - Amazon Redshift ODBC driver v2.1.3
  - Amazon Redshift Python driver v2.1.3
- Amazon Redshift drivers support `IdpTokenAuthPlugin` starting from the following versions:
  - Amazon Redshift JDBC driver v2.1.0.19
  - Amazon Redshift ODBC driver v2.0.0.9
  - Amazon Redshift Python driver v2.0.914

- **No support for enhanced VPC routing** – Enhanced VPC routing isn't supported when you use native identity provider federation. For more information about enhanced VPC routing, see Enhanced VPC routing in Amazon Redshift.
- **AWS IAM Identity Center caching** – AWS IAM Identity Center caches session information. This might cause unpredictable access issues when you attempt to connect to your Redshift database via Redshift query editor v2. This is because the associated AWS IAM Identity Center session in query editor v2 remains valid, even in a case where the database user is signed out of the AWS console. The cache expires after one hour, which typically remediates any issues.

Setting up the identity provider on Amazon Redshift

This section shows the steps to configure the identity provider and Amazon Redshift to establish communication for native identity provider federation. You need an active account with your identity provider. Prior to configuring Amazon Redshift, you register Redshift as an application with your identity provider, granting administrator consent.

Complete the following steps in Amazon Redshift:

1. You run a SQL statement to register the identity provider, including descriptions of the Azure application metadata. To create the identity provider in Amazon Redshift, run the following command after replacing the parameter values *issuer*, *client_id*, *client_secret*, and *audience*. These parameters are specific to Microsoft Azure AD. Replace the identity provider name with a name of your choosing, and replace the namespace with a unique name to contain users and roles from your identity provider directory.

   ```
   CREATE IDENTITY PROVIDER oauth_standard TYPE azure
   NAMESPACE 'aad'
   PARAMETERS '{
   "issuer":"https://sts.windows.net/2sdfdsf-d475-420d-b5ac-667adad7c702/",
   "client_id":"<client_id>",
   "client_secret":"BUAH~ewrqewrqwerUUY^%tHe1oNZShoiU7",
   "audience":["https://analysis.windows.net/powerbi/connector/AmazonRedshift"]
   }'
   ```

   The type `azure` indicates that the provider specifically facilitates communication with Microsoft Azure AD. This is currently the only supported third-party identity provider.

   - *issuer* - The issuer ID to trust when a token is received. The unique identifier for the *tenant_id* is appended to the issuer.

- *client_id* - The unique, public identifier of the application registered with the identity provider. This can be referred to as the application ID.
- *client_secret* - A secret identifier, or password, known only to the identity provider and the registered application.
- *audience* - The Application ID that is assigned to the application in Azure.

Instead of using a shared client secret, you can set parameters to specify a certificate, a private key, and a private key password when you create the identity provider.

```
CREATE IDENTITY PROVIDER example_idp TYPE azure
NAMESPACE 'example_aad'
PARAMETERS '{"issuer":"https://sts.windows.net/2sdfdsf-d475-420d-
b5ac-667adad7c702/",
"client_id":"<client_id>",
"audience":["https://analysis.windows.net/powerbi/connector/AmazonRedshift"],
"client_x5t":"<certificate thumbprint>",
"client_pk_base64":"<private key in base64 encoding>",
"client_pk_password":"test_password"}';
```

The private key password, *client_pk_password*, is optional.

2. Optional: Run SQL commands in Amazon Redshift to pre-create users and roles. This facilitates granting permissions in advance. The role name in Amazon Redshift is like the following: *<Namespace>:<GroupName on Azure AD>*. For example, when you create a group in Microsoft Azure AD called `rsgroup` and a namespace called `aad`, the role name is `aad:rsgroup`. The user and role names in Amazon Redshift are defined from these user names and group memberships in the identity provider namespace.

   The mapping for roles and users includes verifying their `external_id` value, to ensure it's up to date. The external ID maps to the identifier of the group or user in the identity provider. For example, a role's external ID maps to the corresponding Azure AD group ID. Similarly, each user's external ID maps to their ID in the identity provider.

   ```
   create role "aad:rsgroup";
   ```

3. Grant relevant permissions to roles per your requirements. For example:

   ```
   GRANT SELECT on all tables in schema public to role "aad:rsgroup";
   ```

4. You can also grant permissions to a specific user.

```
GRANT SELECT on table foo to aad:alice@example.com
```

   Note that a federated external user's role membership is available only in that user's session. This has implications for creating database objects. When a federated external user creates any view or stored procedure, for instance, the same user can't delegate permission of those objects to other users and roles.

An explanation of namespaces

A namespace maps a user or role to a specific identity provider. For example, the prefix for users created in AWS IAM is `iam:`. This prefix prevents user name collisions and makes support for multiple identity stores possible. If a user alice@example.com from the identity source registered with *aad* namespace logs in, the user `aad:alice@example.com` is created in Redshift if it doesn't already exist. Note that a user and role namespace has a different function than an Amazon Redshift cluster namespace, which is a unique identifier associated with a cluster.

Automatically creating Amazon Redshift roles for identity providers

This feature allows you to automatically create roles in Redshift based on group membership from your Identity Provider (IdP). Auto-creating roles supports the Azure Active Directory with the native IdP integration.

There are several benefits to auto-creating roles. When you auto-create a role, Redshift creates the role with group membership in your IdP, so you can avoid tedious manual role creation and maintenance. You also have the option to filter which groups are mapped to Redshift roles.

How it works

When you, as an IdP user, log into Redshift, the following sequence of events happen:

1. Redshift retrieves your group memberships from the IdP.
2. Redshift automatically creates roles mapping to those groups, with the role format *`idp_namespace`*`:`*`rolename`*.
3. Redshift grants you permissions with the mapped roles.

Upon each user login, each group that's not present in catalog but that the user is part of, is auto-created. You can optionally set include and exclude filters to control which IdP groups have Redshift roles created.

Configuring auto-create roles

Use the `CREATE IDENTITY PROVIDER` and `ALTER IDENTITY PROVIDER` commands to enable and configure automatic role creation.

```
-- Create a new IdP with auto role creation enabled
CREATE IDENTITY PROVIDER <idp_name> TYPE azure
  NAMESPACE '<namespace>'
  APPLICATION_ARN 'app_arn'
  IAM_ROLE 'role_arn'
  AUTO_CREATE_ROLES TRUE;

-- Enable on existing IdP
ALTER IDENTITY PROVIDER <idp_name>
  AUTO_CREATE_ROLES TRUE;

-- Disable
ALTER IDENTITY PROVIDER <idp_name>
  AUTO_CREATE_ROLES FALSE;
```

Filtering groups

You can optionally filter which IdP groups are mapped to Redshift roles using `INCLUDE` and `EXCLUDE` patterns. When patterns conflict, `EXCLUDE` takes precedence over `INCLUDE`.

```
-- Only create roles for groups with 'dev'
CREATE IDENTITY PROVIDER <idp_name> TYPE azure
  ...
  AUTO_CREATE_ROLES TRUE
  INCLUDE GROUPS LIKE '%dev%';

-- Exclude 'test' groups
ALTER IDENTITY PROVIDER <idp_name>
  AUTO_CREATE_ROLES TRUE
  EXCLUDE GROUPS LIKE '%test%';
```

Examples

The following example shows how to turn on auto-create roles with no filtering.

```
CREATE IDENTITY PROVIDER prod_idc TYPE azure ...
  AUTO_CREATE_ROLES TRUE;
```

The following example includes development groups and excludes test groups.

```
ALTER IDENTITY PROVIDER prod_idc
  AUTO_CREATE_ROLES TRUE
  INCLUDE GROUPS LIKE '%dev%'
  EXCLUDE GROUPS LIKE '%test%';
```

Best practices

Consider the following best practives when you enable auto-create for roles:

- Use INCLUDE and EXCLUDE filters to control which groups get roles.
- Periodically audit roles and clean up unused ones.
- Leverage Redshift role hierarchies to simplify permission management.

Connect Redshift with AWS IAM Identity Center for a single sign-on experience

You can manage user and group access to Amazon Redshift data warehouses through trusted-identity propagation.

Trusted identity propagation is an AWS IAM Identity Center feature that administrators of connected AWS services can use to grant and audit access to service data. Access to this data is based on user attributes such as group associations. Setting up trusted identity propagation requires collaboration between the administrators of connected AWS services and the IAM Identity Center administrators. For more information, see Prerequisites and considerations.

To illustrate one end-to-end case, you can use an Amazon Quick dashboard or Amazon Redshift query editor v2 to access Redshift. Access in this case is based on AWS IAM Identity Center groups. Redshift can determine who a user is and their group memberships. AWS IAM Identity Center also makes it possible to connect and manage identities through a third-party identity provider (IdP) like Okta or PingOne.

After your administrator sets up the connection between Redshift and AWS IAM Identity Center, they can configure fine-grained access based on identity-provider groups to authorize user access to data.

Important

When you delete a user from an AWS IAM Identity Center or a connected identity provider (IdP) directory, the user is not automatically deleted from the Amazon Redshift catalog. To manually delete the user from the Amazon Redshift catalog, run the `DROP USER` command to fully delete the user that was removed from an AWS IAM Identity Center or IdP. For more information about how to drop a user, see DROP USER in the *Amazon Redshift Database Developer Guide*.

Benefits of Redshift integration with AWS IAM Identity Center

Using AWS IAM Identity Center with Redshift can benefit your organization in the following ways:

- Dashboard authors in Amazon Quick can connect to Redshift data sources without having to re-enter passwords or requiring an administrator to set up IAM roles with complex permissions.
- AWS IAM Identity Center provides a central location for your workforce users in AWS. You can create users and groups directly in AWS IAM Identity Center or connect existing users and groups that you manage in a standards-based identity provider like Okta, PingOne, or Microsoft Entra ID (Azure AD). AWS IAM Identity Center directs authentication to your chosen source of truth for users and groups, and it maintains a directory of users and groups for access by Redshift. For more information, see Manage your identity source and Supported identity providers in the *AWS IAM Identity Center User Guide*.
- You can share one AWS IAM Identity Center instance with multiple Redshift clusters and workgroups with a simple auto-discovery and connect capability. This makes it fast to add clusters without the extra effort of configuring the AWS IAM Identity Center connection for each, and it ensures that all clusters and workgroups have a consistent view of users, their attributes, and groups. Note that your organization's AWS IAM Identity Center instance must be in the same region as any Redshift datashares you're connecting to.
- Because user identities are known and logged along with data access, it's easier for you to meet compliance regulations through auditing user access in AWS CloudTrail.

Administrator personas for connecting applications

The following are personas that are key to connecting analytics applications to the AWS IAM Identity Center managed application for Redshift:

- **Application administrator** – Creates an application and configures which services it will enable identity-token exchanges with. This administrator also specifies which users or groups have access to the application.
- **Data administrator** – Configures fine-grained access to data. Users and groups in AWS IAM Identity Center can map to specific permissions.

Connecting to Amazon Redshift with AWS IAM Identity Center through Amazon Quick

The following shows how to use Quick to authenticate with Redshift when it's connected to and access is managed through AWS IAM Identity Center: Authorizing connections from Quick to Amazon Redshift clusters. These steps apply to Amazon Redshift Serverless too.

Connecting to Amazon Redshift with AWS IAM Identity Center through Amazon Redshift query editor v2

Upon completing the steps to set up an AWS IAM Identity Center connection with Redshift, the user can access the database and appropriate objects in the database through their AWS IAM Identity Center-based, namespace-prefixed identity. For more information about connecting to Redshift databases with query editor v2 sign-in, see Querying a database using the query editor v2.

Using AWS IAM Identity Center across multiple AWS Regions

Amazon Redshift supports AWS IAM Identity Center in multiple AWS Regions. You can extend AWS IAM Identity Center from your primary AWS Region to additional Regions for improved performance through proximity to users and reliability. When a new Region is added in AWS IAM Identity Center, you can create Redshift IAM Identity Center applications in the new Region without replicating identities from the primary Region. You can set up Amazon Redshift federated permissions using AWS IAM Identity Center in the new Region where you can enable row-level, column-level, and masking controls. For more details to get started with AWS IAM Identity Center in multiple Regions, see Manage AWS IAM Identity Center in multiple AWS Regions in the *AWS IAM Identity Center User Guide*.

Setting up AWS IAM Identity Center integration with Amazon Redshift

Your Amazon Redshift cluster administrator or Amazon Redshift Serverless administrator must perform several steps to configure Redshift as an AWS IAM Identity Center enabled application. This makes it so Redshift can discover and connect to AWS IAM Identity Center automatically to receive sign-in and user directory services. After this, when your Redshift administrator creates a

cluster or workgroup, they can enable the new data warehouse to use AWS IAM Identity Center to manage database access.

The point of enabling Redshift as an AWS IAM Identity Center managed application is so you can control user and group permissions from within AWS IAM Identity Center, or from a third-party identity provider that's integrated with it. When your database users sign in to a Redshift database, for example an analyst or a data scientist, it checks their groups in AWS IAM Identity Center and these match up with role names in Redshift. In this manner, a group that defines the name for a Redshift database role can access a set of tables for sales analytics, for example. The sections that follow show how to set this up.

Prerequisites

These are the prerequisites for integrating AWS IAM Identity Center with Amazon Redshift:

- *Account configuration* – You must configure AWS IAM Identity Center in your AWS organization's management account if you plan to have cross-account use cases, or if you use Redshift clusters in different accounts with the same AWS IAM Identity Center instance. This includes configuring your identity source. For more information, see Getting Started, workforce identities, and supported identity providers in the *AWS IAM Identity Center User Guide*. You must ensure that you have created users or groups in AWS IAM Identity Center, or synchronized users and groups from your identity source before you can assign them to data in Redshift.

  > **Note**
  >
  > You have an option to use an account instance of AWS IAM Identity Center, provided that Redshift and AWS IAM Identity Center are in the same account. You can create this instance using a widget when you create and configure a Redshift cluster or workgroup.

- *Configuring a trusted token issuer* – In some cases, you may need to use a trusted token issuer, which is an entity that can issue and verify trust tokens. Before you can do so, preliminary steps are required before the Redshift administrator who configures AWS IAM Identity Center integration can select the trusted token issuer and add the necessary attributes to complete the configuration. This can include configuring an external identity provider to serve as a trusted token issuer and adding its attributes in the AWS IAM Identity Center console. To complete these steps, see Using applications with a trusted token issuer.

> **Note**
>
> Setting up a trusted token issuer isn't required for all external connections. Connecting to your Redshift database with Amazon Redshift query editor v2 doesn't require trusted-token issuer configuration. But it can apply for third-party applications such as dashboards or custom applications that authenticate with your identity provider.

- *Configuring an IAM role or roles* – The sections that follow mention permissions that must be configured. You will have to add permissions per IAM best practices. Specific permissions are detailed in the procedures that follow.
- *Network configuration for enhanced VPC routing* – If enhanced VPC routing is turned on, your VPC must be able to reach the AWS IAM Identity Center services. Interface VPC endpoints are the recommended way to provide this connectivity.

> **Note**
>
> Set up this connectivity before users sign in. For more information about using enhanced VPC routing with AWS IAM Identity Center, see Using AWS IAM Identity Center authentication with enhanced VPC routing.

For more information, see Getting Started with AWS IAM Identity Center.

Configuring your identity provider to work with AWS IAM Identity Center

The first step in controlling user and group identity management is to connect to AWS IAM Identity Center and configure your identity provider. You can use AWS IAM Identity Center itself as your identity provider, or you can connect a third-party identity store, such as Okta, for instance. For more information about setting up the connection to and configuring your identity provider, see Connect to an external identity provider in the *AWS IAM Identity Center user guide*. Make sure at the end of this process that you have a small collection of users and groups added to AWS IAM Identity Center, for test purposes.

Administrative Permissions

Permissions required for Redshift/AWS IAM Identity Center application lifecycle management

You must create an IAM identity, which a Redshift administrator uses to configure Redshift for use with AWS IAM Identity Center. Most commonly, you would create an IAM role with permissions and assign it to other identities as required. It must have the permissions listed to perform the following actions.

Creating the Redshift/AWS IAM Identity Center application

- `sso:PutApplicationAssignmentConfiguration` – For security.
- `sso:CreateApplication` – Used to create an AWS IAM Identity Center application.
- `sso:PutApplicationAuthenticationMethod` – Grants Redshift authentication access.
- `sso:PutApplicationGrant` – Used to change the trusted token issuer information.
- `sso:PutApplicationAccessScope` – For Redshift AWS IAM Identity Center application setup. This includes for AWS Lake Formation and for Amazon S3 Access Grants.
- `redshift:CreateRedshiftIdcApplication` – Used to create the Redshift AWS IAM Identity Center application.

Describing the Redshift/AWS IAM Identity Center application

- `sso:GetApplicationGrant` – Used to list trusted token issuer information.
- `sso:ListApplicationAccessScopes` – For Redshift AWS IAM Identity Center application setup to list downstream integrations, such as for AWS Lake Formation and S3 Access Grants.
- `redshift:DescribeRedshiftIdcApplications` – Used to describe existing AWS IAM Identity Center applications.

Changing the Redshift/AWS IAM Identity Center application

- `redshift:ModifyRedshiftIdcApplication` – Used to change an existing Redshift application.
- `sso:UpdateApplication` – Used to update an AWS IAM Identity Center application.
- `sso:GetApplicationGrant` – Gets the trust token issuer information.
- `sso:ListApplicationAccessScopes` – For Redshift AWS IAM Identity Center application setup.

- `sso:DeleteApplicationGrant` – Deletes the trust token issuer information.
- `sso:PutApplicationGrant` – Used to change the trusted token issuer information.
- `sso:PutApplicationAccessScope` – For Redshift AWS IAM Identity Center application setup. This includes for AWS Lake Formation and for Amazon S3 Access Grants.
- `sso:DeleteApplicationAccessScope` – For deleting Redshift AWS IAM Identity Center application setup. This includes for AWS Lake Formation and for Amazon S3 Access Grants.

Deleting the Redshift/AWSIAM Identity Center application

- `sso:DeleteApplication` – Used to delete an AWS IAM Identity Center application.
- `redshift:DeleteRedshiftIdcApplication` – Gives the ability to delete an existing Redshift AWS IAM Identity Center application.

Permissions required for Redshift/query editor v2 application lifecycle management

You must create an IAM identity, which a Redshift administrator uses to configure Redshift for use with AWS IAM Identity Center. Most commonly, you would create an IAM role with permissions and assign it to other identities as required. It must have the permissions listed to perform the following actions.

Creating the query editor v2 application

- `redshift:CreateQev2IdcApplication` – Used to create the QEV2 application.
- `sso:CreateApplication` – Gives the ability to create an AWS IAM Identity Center application.
- `sso:PutApplicationAuthenticationMethod` – Grants Redshift authentication access.
- `sso:PutApplicationGrant` – Used to change the trusted token issuer information.
- `sso:PutApplicationAccessScope` – For Redshift AWS IAM Identity Center application setup. This includes query editor v2.
- `sso:PutApplicationAssignmentConfiguration` – For security.

Describe the query editor v2 application

- `redshift:DescribeQev2IdcApplications` – Used to describe the AWS IAM Identity Center QEV2 application.

Change the query editor v2 application

- `redshift:ModifyQev2IdcApplication` – Used to change the AWS IAM Identity Center QEV2 application.
- `sso:UpdateApplication` – Used to change the AWS IAM Identity Center QEV2 application.

Delete the query editor v2 application

- `redshift:DeleteQev2IdcApplication` – Used to delete the QEV2 application.
- `sso:DeleteApplication` – Used to delete the QEV2 application.

> **Note**
>
> In the Amazon Redshift SDK, the following APIs aren't available:
>
> - CreateQev2IdcApplication
> - DescribeQev2IdcApplications
> - ModifyQev2IdcApplication
> - DeleteQev2IdcApplication
>
> These actions are specific to performing AWS IAM Identity Center integration with Redshift QEV2 in the AWS console. For more information, see Actions defined by Amazon Redshift.

Permissions required for the database administrator to connect new resources in the console

These permissions are required to connect new provisioned clusters or Amazon Redshift Serverless workgroups during the creation process. If you have these permissions, a selection appears in the console to choose to connect to the AWS IAM Identity Center managed application for Redshift.

- `redshift:DescribeRedshiftIdcApplications`
- `sso:ListApplicationAccessScopes`
- `sso:GetApplicationAccessScope`
- `sso:GetApplicationGrant`

As a best practice, we recommend attaching permissions policies to an IAM role and then assigning it to users and groups as needed. For more information, see Identity and access management in Amazon Redshift.

Setting up Redshift as an AWS managed application with AWS IAM Identity Center

Before AWS IAM Identity Center can manage identities for an Amazon Redshift provisioned cluster or an Amazon Redshift Serverless workgroup, the Redshift administrator must complete the steps to make Redshift an AWS IAM Identity Center managed application:

1. Select **AWS IAM Identity Center integration** in the Amazon Redshift or Amazon Redshift Serverless console menu, and then select **Connect to AWS IAM Identity Center**. From there you step through a series of selections to populate the properties for AWS IAM Identity Center integration.
2. Choose a display name and a unique name for Redshift's AWS IAM Identity Center-managed application.
3. Specify the namespace for your organization. This is typically an abbreviated version of your organization's name. It's added as a prefix for your AWS IAM Identity Center-managed users and roles in the Redshift database.
4. Select an IAM role to use. This IAM role should be separate from others used for Redshift, and we recommend that it isn't used for other purposes. The specific policy permissions required are the following:
   - `sso:DescribeApplication` – Required to create an identity provider (IdP) entry in the catalog.
   - `sso:DescribeInstance` – Used to manually create IdP federated roles or users.
5. Configure client connections and trusted token issuers. Configuring trusted token issuers facilitates trusted identity propagation by setting up a relationship with an external identity provider. Identity propagation makes it possible for a user, for example, to sign into one application and access specific data in another application. This allows users to gather data from disparate locations more seamlessly. At this step, in the console, you set attributes for each trusted token issuer. The attributes include the name and the audience claim (or *aud claim*), which you might have to get from the tool's or service's configuration attributes. You might also need to supply the application name from the third-party tool's JSON Web Token (JWT).

> **Note**
>
> The `aud claim` required from each third-party tool or service can vary, based on the token type, which can be an access token issued by an identity provider, or another type, like an ID token. Each vendor can be different. When you're implementing trusted-identity propagation and integrating with Redshift, it's required to supply the correct *aud* value for the token type that the third-party tool sends to AWS. Check the recommendations of your tool or service vendor.

For detailed information regarding trusted-identity propagation, see Trusted identity propagation overview in the *AWS IAM Identity Center User Guide*.

After the Redshift administrator finishes the steps and saves the configuration, the AWS IAM Identity Center properties appear in the Redshift console. You can also query the system view SVV_IDENTITY_PROVIDERS to verify the application's properties. These include the application name and the namespace. You use the namespace as a prefix for Redshift database objects that are associated with the application. Completing these tasks makes Redshift an AWS IAM Identity Center enabled application. The properties in the console include the integration status. It says **Enabled** when the integration is completed. After this process, AWS IAM Identity Center integration can be enabled on each new cluster.

After configuration, you can include users and groups from AWS IAM Identity Center in Redshift by choosing the **Users** or **Groups** tab and choosing **Assign**.

Enabling AWS IAM Identity Center integration for a new Amazon Redshift cluster or Amazon Redshift Serverless workgroup

Your database administrator configures new Redshift resources to work in alignment with AWS IAM Identity Center to make sign-in and data access easier. This is performed as part of the steps to create a provisioned cluster or a Serverless workgroup. Anyone with permissions to create Redshift resources can perform these AWS IAM Identity Center integration tasks.When you create a provisioned cluster, you start by choosing **Create Cluster** in the Amazon Redshift console. The steps that follow show how to enable AWS IAM Identity Center management for a database. (It doesn't include all of the steps to create a cluster.)

1. Choose **Enable for <your cluster name>** in the section for **IAM Identity Center integration** in the create-cluster steps.
2. There's a step in the process when you enable integration. You do this by choosing **Enable IAM Identity Center integration** in the console.
3. For the new cluster or workgroup, create database roles in Redshift using SQL commands. The following is the command:

   ```
   CREATE ROLE <idcnamespace:rolename>;
   ```

   The namespace and role name are the following:

   - *IAM Identity Center namespace prefix* – This is the namespace you defined when you set up the connection between AWS IAM Identity Center and Redshift.
   - *Role name* – This Redshift database role must match the group name in AWS IAM Identity Center.

   Redshift connects with AWS IAM Identity Center and fetches the information needed to create and map the database role to the AWS IAM Identity Center group.

Note that when a new data warehouse is created, the IAM role specified for AWS IAM Identity Center integration is automatically attached to the provisioned cluster or Amazon Redshift Serverless workgroup. After you finish entering the required cluster metadata and create the resource, you can check the status for AWS IAM Identity Center integration in the properties. If your group names in AWS IAM Identity Center have spaces, it's required to use quotes in SQL when you create the matching role.

After you enable the Redshift database and create roles, you are ready to connect to the database with Amazon Redshift query editor v2 or Amazon Quick. The details are explained further in sections that follow.

Setting up the default `RedshiftIdcApplication` using the API

Setup is performed by your identity administrator. Using the API, you create and populate a `RedshiftIdcApplication`, which represents the Redshift application within AWS IAM Identity Center.

1. To start, you can create users and add them to groups in AWS IAM Identity Center. You do this in the AWS console for AWS IAM Identity Center.
2. Call `create-redshift-idc-application` to create an AWS IAM Identity Center application and make it compatible with Redshift usage. You create the application by populating the required values. The display name is the name to display on the AWS IAM Identity Center dashboard. The IAM role ARN is an ARN that has permissions to AWS IAM Identity Center and is also assumable by Redshift.

```
aws redshift create-redshift-idc-application
--idc-instance-arn 'arn:aws:sso:::instance/ssoins-1234a01a1b12345d'
--identity-namespace 'MYCO'
--idc-display-name 'TEST-NEW-APPLICATION'
--iam-role-arn 'arn:aws:redshift:us-east-1:012345678901:role/TestRedshiftRole'
--redshift-idc-application-name 'myredshiftidcapplication'
```

   The following example shows a sample `RedshiftIdcApplication` response that's returned from the call to `create-redshift-idc-application`.

```
"RedshiftIdcApplication": {
                "IdcInstanceArn": "arn:aws:sso:::instance/ssoins-1234a01a1b12345d",
                "RedshiftIdcApplicationName": "test-application-1",
                "RedshiftIdcApplicationArn": "arn:aws:redshift:us-
east-1:012345678901:redshiftidcapplication:12aaa111-3ab2-3ab1-8e90-b2d72aea588b",
                "IdentityNamespace": "MYCO",
                "IdcDisplayName": "Redshift-Idc-Application",
                "IamRoleArn": "arn:aws:redshift:us-east-1:012345678901:role/
TestRedshiftRole",
                "IdcManagedApplicationArn": "arn:aws:sso::012345678901:application/
ssoins-1234a01a1b12345d/apl-12345678910",
                "IdcOnboardStatus": "arn:aws:redshift:us-
east-1:123461817589:redshiftidcapplication",
                "RedshiftIdcApplicationArn": "Completed",
                "AuthorizedTokenIssuerList": [
                       "TrustedTokenIssuerArn": ...,
                       "AuthorizedAudiencesList": [...]...
                ]}
```

3. You can use `create-application-assignment` to assign particular groups or individual users to the managed application in AWS IAM Identity Center. By doing this, you can specify groups to manage through AWS IAM Identity Center. If the database administrator creates

database roles in Redshift, group names in AWS IAM Identity Center map to the role names in Redshift. The roles control permissions in the database. For more information, see Assign user access to applications in the AWS IAM Identity Center console.

4. After you enable the application, call `create-cluster` and include the Redshift managed application ARN from AWS IAM Identity Center. Doing this associates the cluster with the managed application in AWS IAM Identity Center.

Associating an AWS IAM Identity Center application with an existing cluster or workgroup

If you have an existing cluster or workgroup that you would like to enable for AWS IAM Identity Center integration, it is possible to do so, running SQL commands. You can also run SQL commands to change settings for the integration. For more information, see ALTER IDENTITY PROVIDER.

It's also possible to drop an existing identity provider. The following example shows how CASCADE deletes users and roles attached to the identity provider.

```
DROP IDENTITY PROVIDER
<provider_name> [ CASCADE ]
```

Setting up user permissions

An administrator configures permissions to various resources, based on users' identity attributes and group memberships, within their identity provider or within AWS IAM Identity Center directly.For example, the identity-provider administrator can add a database engineer to a group appropriate to their role. This group name maps to a Redshift database role name. The role provides or restricts access to specific tables or views in Redshift.

Using AWS IAM Identity Center authentication with enhanced VPC routing

When enhanced VPC routing is turned on, Redshift routes traffic through your virtual private cloud (VPC) instead of over the internet. This includes the calls that Redshift makes to authenticate an AWS IAM Identity Center user. This applies whether the user signs in interactively or connects with an AWS IAM Identity Center token obtained through trusted identity propagation. For these connections to succeed, Redshift must be able to reach AWS IAM Identity Center from your VPC. Interface VPC endpoints are the recommended way to provide this connectivity. They let your provisioned cluster or Amazon Redshift Serverless workgroup reach AWS IAM Identity Center over the AWS network.

Important

These requirements apply to any connection authenticated through AWS IAM Identity Center. IAM users and local database users authenticate through different mechanisms and aren't affected. If enhanced VPC routing is turned off, these endpoints aren't required.

Note

Changing enhanced VPC routing restarts a provisioned cluster. For more information about turning on enhanced VPC routing, see Turning on enhanced VPC routing.

Interface VPC endpoints

Create the following interface VPC endpoints (AWS PrivateLink) in the VPC where your cluster or workgroup resides. Both endpoints are required. If either endpoint is missing or unreachable, AWS IAM Identity Center sign-in fails.

`com.amazonaws.region.sso-oauth`

Validates the AWS IAM Identity Center access token that the user presents at sign-in and exchanges it for a token scoped to your Redshift session.

`com.amazonaws.region.identitystore`

Resolves the user and their AWS IAM Identity Center group memberships from the AWS IAM Identity Center identity store.

For information about creating an interface VPC endpoint, see Create a VPC endpoint in the *Amazon VPC User Guide*.

Before you create the endpoints, confirm that the `DNS hostnames` and `DNS resolution` attributes are turned on for the VPC where your cluster or workgroup runs. Private DNS names depend on both attributes. For more information, see DNS attributes for your VPC in the *Amazon VPC User Guide*.

Enhanced VPC routing requires that your provisioned cluster or Redshift Serverless workgroup is not publicly accessible.

When you create each endpoint, do the following:

- **Turn on private DNS names** – Redshift connects to these services using their standard DNS names. Without private DNS names, those names continue to resolve to public addresses and the calls don't route to your endpoints.
- **Create the endpoints in the AWS IAM Identity Center Region** – AWS IAM Identity Center is a Regional service. Create these endpoints in the Region where your AWS IAM Identity Center instance runs. If your cluster or workgroup runs in a different Region and you don't use AWS IAM Identity Center multi-Region replication, use the cross-Region endpoint option so that the endpoints reach the AWS IAM Identity Center Region.
- **Choose subnets in the same Availability Zones** – Select subnets in the Availability Zones that your cluster or workgroup uses.
- **Allow inbound HTTPS** – The security group attached to each endpoint must allow inbound traffic on TCP port 443 from your cluster or workgroup subnets.
- **Keep the default endpoint policy** – The default policy allows full access. If your organization requires a restrictive endpoint policy, scope it by principal rather than by a list of actions. Then verify that sign-in still succeeds.

Using AWS Network Firewall

AWS Network Firewall is optional. If you filter outbound traffic with AWS Network Firewall instead of using interface VPC endpoints, the firewall policy must allow the AWS IAM Identity Center service domains. Otherwise sign-in fails when Redshift can't reach them. Add the following domains to your firewall's allow list, replacing *region* with your AWS IAM Identity Center Region:

`oidc.`*region*`.amazonaws.com`

Validates the AWS IAM Identity Center access token and exchanges it for a token scoped to your Redshift session.

`.sso.`*region*`.amazonaws.com`

Reaches the AWS IAM Identity Center service to resolve the AWS IAM Identity Center application and its assignments.

`identitystore.`*region*`.amazonaws.com`

Resolves the user and their AWS IAM Identity Center group memberships.

If your firewall uses a deny-by-default policy, it must also allow the domains that Redshift uses for general operation under enhanced VPC routing, such as Amazon S3 and, if you query a data lake, AWS Glue. For more information, see Enhanced VPC routing in Amazon Redshift.

Using interface VPC endpoints, as described in Interface VPC endpoints, keeps this traffic on the AWS network and doesn't require firewall allow-listing.

Automatically creating Amazon Redshift roles for AWS IAM Identity Center

This feature is an integration with AWS IAM Identity Center that allows you to automatically create roles in Redshift based on group membership.

There are several benefits to auto-creating roles. When you auto-create a role, Redshift creates the role with group membership in your IdP, so you can avoid tedious manual role creation and maintenance. You also have the option to filter which groups are mapped to Redshift roles with include and exclude patterns.

How it works

When you, as an IdP user, log into Redshift, the following sequence of events happen:

1. Redshift retrieves your group memberships from the IdP.
2. Redshift automatically creates roles mapping to those groups, with the role format *idp_namespace*:*rolename*.
3. Redshift grants you permissions with the mapped roles.

Upon each user login, each group that's not present in catalog but that the user is part of, is auto-created. You can optionally set include and exclude filters to control which IdP groups have Redshift roles created.

Configuring auto-create roles

Use the `CREATE IDENTITY PROVIDER` and `ALTER IDENTITY PROVIDER` commands to enable and configure automatic role creation.

```
-- Create a new IdP with auto role creation enabled
CREATE IDENTITY PROVIDER <idp_name> TYPE AWSIDC
  NAMESPACE '<namespace>'
  APPLICATION_ARN 'app_arn'
  IAM_ROLE 'role_arn'
  AUTO_CREATE_ROLES TRUE;
```

```
-- Enable on existing IdP
ALTER IDENTITY PROVIDER <idp_name>
  AUTO_CREATE_ROLES TRUE;

-- Disable
ALTER IDENTITY PROVIDER <idp_name>
  AUTO_CREATE_ROLES FALSE;
```

Filtering groups

You can optionally filter which IdP groups are mapped to Redshift roles using `INCLUDE` and `EXCLUDE` patterns. When patterns conflict, `EXCLUDE` takes precedence over `INCLUDE`.

```
-- Only create roles for groups with 'dev'
CREATE IDENTITY PROVIDER <idp_name> TYPE AWSIDC
  ...
  AUTO_CREATE_ROLES TRUE
  INCLUDE GROUPS LIKE '%dev%';

-- Exclude 'test' groups
ALTER IDENTITY PROVIDER <idp_name>
  AUTO_CREATE_ROLES TRUE
  EXCLUDE GROUPS LIKE '%test%';
```

Examples

The following example shows how to turn on auto-create roles with no filtering.

```
CREATE IDENTITY PROVIDER prod_idc TYPE AWSIDC  ...
  AUTO_CREATE_ROLES TRUE;
```

The following example includes development groups and excludes test groups.

```
ALTER IDENTITY PROVIDER prod_idc
  AUTO_CREATE_ROLES TRUE
  INCLUDE GROUPS LIKE '%dev%'
  EXCLUDE GROUPS LIKE '%test%';
```

Best practices

Consider the following best practives when you enable auto-create for roles:

- Use `INCLUDE` and `EXCLUDE` filters to control which groups get roles.
- Periodically audit roles and clean up unused ones.
- Leverage Redshift role hierarchies to simplify permission management.

Amazon Redshift Integration with Amazon S3 Access Grants

Using integration with Amazon S3 Access Grants, you can seamlessly propagate your IAM Identity Center identities to control access to Amazon S3 data. This integration lets you authorize Amazon S3 data access based on IAM Identity Center users and groups.

For information about Amazon S3 Access Grants, see Managing access with S3 Access Grants.

Using Amazon S3 Access Grants gives your application the following benefits:

- Fine-grained access control to Amazon S3 data, based on IAM Identity Center identities.
- Centralized management of IAM Identity Center identities across Amazon Redshift and Amazon S3.
- You can avoid managing separate IAM permissions for Amazon S3 access.

How it works

To integrate your application with Amazon S3 access grants, you do the following:

- First, you configure Amazon Redshift to integrate with Amazon S3 Access Grants using the AWS Management Console or AWS CLI.
- Next, a user with IdC administrator privileges grants Amazon S3 bucket or prefix access to specific IdC users/groups, using the Amazon S3 Access Grants service. For more information, see Working with grants in S3 Access Grants.
- When an IdC user authenticated to Redshift runs a query accessing S3 (such as a COPY, UNLOAD, or Spectrum operation), Amazon Redshift retrieves temporary S3 access credentials scoped to that IdC identity from the Amazon S3 Access Grants service.
- Amazon Redshift then uses the retrieved temporary credentials to access the authorized Amazon S3 locations for that query.

Setting up integration with Amazon S3 Access Grants

To set up integration with integration with Amazon S3 Access Grants for Amazon Redshift, do the following:

Topics

- Setting up integration with Amazon S3 Access Grants using the AWS Management Console
- Enabling integration with Amazon S3 Access Grants using the AWS CLI

Setting up integration with Amazon S3 Access Grants using the AWS Management Console

1. Open the Amazon Redshift console.
2. Choose your cluster from the **Clusters** pane.
3. In your cluster's details page, in the **Identity provider integration** section, enable integration with the **S3 Access Grants** service.

> **Note**
>
> The **Identity provider integration** section doesn't appear if you don't have IAM Identity Center configured. For more information, see Enabling AWS IAM Identity Center.

Enabling integration with Amazon S3 Access Grants using the AWS CLI

1. To create a new Amazon Redshift IdC application with S3 integration enabled, do the following:

```
aws redshift create-redshift-idc-application <other parameters>
  --service-integrations '[ {"S3AccessGrants": [{"ReadWriteAccess":
 {"Authorization": "Enabled"}}]} ]'
```

2. To modify an existing application to enable S3 Access Grants integration, do the following:

```
aws redshift modify-redshift-idc-application <other parameters>
  --service-integrations '[ {"S3AccessGrants": [{"ReadWriteAccess":
 {"Authorization": "Enabled"}}]} ]'
```

3. To modify an existing application to disable S3 Access Grants integration, do the following:

```
aws redshift modify-redshift-idc-application <other parameters>
  --service-integrations '[ {"S3AccessGrants": [{"ReadWriteAccess":
 {"Authorization": "Disabled"}}]} ]'
```

Using integration with S3 Access Grants

After you configure S3 Access Grants integration, queries that access S3 data (such as COPY, UNLOAD, or Spectrum queries) use the IdC identity for authorization. Users who are not authenticated using IdC can also run these queries, but those user accounts don't take advantage of the centralized administration that IdC provides.

The following example shows queries that run with S3 Access Grants integration:

```
COPY table FROM 's3://mybucket/data';  // -- Redshift uses IdC identity
UNLOAD ('SELECT * FROM table') TO 's3://mybucket/unloaded/'    // -- Redshift uses IdC
 identity
```

Querying data through AWS Lake Formation

Using AWS Lake Formation makes it easier to centrally govern and secure your data lake, and to provide data access. Configuring identity propagation to Lake Formation through AWS IAM Identity Center and Redshift makes it so an administrator can allow fine-grained access to an Amazon S3 data lake, based on the organization's identity-provider (IdP) groups. These groups are managed through AWS IAM Identity Center. This section shows how to configure a couple use cases, querying from a data lake and querying from a data share, that demonstrate how to leverage AWS IAM Identity Center with Redshift to connect to Lake Formation-governed resources.

Using an AWS IAM Identity Center and Redshift connection to query a data lake

These steps cover a use case where you use AWS IAM Identity Center connected to Redshift to query a data lake that's governed by Lake Formation.

Prerequisites

This procedure has several prerequisite steps:

1. AWS IAM Identity Center must be set up to support authentication and identity management with Redshift. You can enable AWS IAM Identity Center from the console and select an identity-provider (IdP) source. After this, synchronize a set of your IdP users with AWS IAM Identity

Center. You must also set up a connection between AWS IAM Identity Center and Redshift, following the steps detailed previously in this document.

2. Create a new Amazon Redshift cluster and enable identity management through AWS IAM Identity Center in the configuration steps.
3. Create a managed AWS IAM Identity Center application for Lake Formation and configure it. This follows setting up the connection between AWS IAM Identity Center and Redshift. The steps are the following:
   a. In the AWS CLI, use the `modify-redshift-idc-application` command to enable the Lake Formation service integration with the AWS IAM Identity Center managed application for Redshift. This call includes the `service-integrations` parameter, which is set to a configuration string value that enables authorization to Lake Formation.
   b. Configure Lake Formation by using the `create-lake-formation-identity-center-configuration` command. This creates an AWS IAM Identity Center application for Lake Formation, which is visible in the AWS IAM Identity Center portal. The administrator must set the `--cli-input-json` argument, whose value is the path to a JSON file that uses the standard format for all AWS CLI API calls. You must include values for the following:
      - `CatalogId` – The Lake Formation catalog ID.
      - `InstanceArn` – The AWS IAM Identity Center instance ARN value.

After the administrator completes the prerequisite configuration, the database administrator can create an external schema for the purpose of querying the data lake.

1. **The administrator creates the external schema** – The Redshift database administrator connects to the database and creates an external schema, using the following SQL statement:

```
CREATE EXTERNAL SCHEMA if not exists my_external_schema from DATA CATALOG database
 'my_lf_integrated_db' catalog_id '12345678901234';
```

   Note that specifying an IAM role isn't required in this case, because access is managed through AWS IAM Identity Center.
2. **The administrator grants permissions** – The administrator grants usage to an AWS IAM Identity Center group, which grants permissions on Redshift resources. This is done by running a SQL statement like the following:

```
GRANT USAGE ON SCHEMA "my_external_schema" to "MYCO:sales";
```

Subsequently, the administrator grants Lake Formation permissions on objects, based on requirements for the organization, using the AWS CLI:

```
aws lakeformation grant-permissions ...
```

3. **Users run queries** – At this point, an AWS IAM Identity Center user that's part of the sales group, for illustration purposes, can log in via query editor v2 to the Redshift database. Then they can run a query that accesses a table in the external schema, like the following sample:

```
SELECT * from my_external_schema.table1;
```

Using an AWS IAM Identity Center and Redshift connection to connect to a datashare

You can access a datashare from a different Redshift data warehouse when access is managed through AWS IAM Identity Center. To do this, you run a query to set up an external database. Prior to completing these steps, it's assumed that you have a connection set up between Redshift and AWS IAM Identity Center, and you've created the AWS Lake Formation application, as detailed in the previous procedure.

1. **Creating the external database** – The administrator creates an external database for data sharing, referencing it through its ARN. The following is a sample that shows how to do it:

```
CREATE DATABASE "redshift_external_db" FROM ARN 'arn:aws:glue:us-
east-1:123456789012:database/redshift_external_db-iad' WITH NO DATA CATALOG SCHEMA;
```

   In this use case, where you are using AWS IAM Identity Center with Redshift for identity management, the IAM role isn't included.

2. **The admin sets up permissions** – After creating a database, the administrator grants usage to an AWS IAM Identity Center group. This grants permissions on Redshift resources:

```
GRANT USAGE ON DATABASE "my_external_db" to "MYCO:sales";
```

   The administrator also grants Lake Formation permissions on objects, using the AWS CLI:

```
aws lakeformation grant-permissions ...
```

3. **Users run queries** – A user from the sales group can query a table in the database, based on the permissions assigned:

```
select * from redshift_external_db.public.employees;
```

For more information about granting permissions on a data lake and granting permissions on data shares, see Granting permissions to users and groups. For more information about granting usage to a schema or to a database, see GRANT.

Integrating your application or tool with OAuth using a trusted token issuer

You can add functionality to client tools you create to connect to Redshift by means of the AWS IAM Identity Center connection. If you already configured Redshift integration to AWS IAM Identity Center, use the properties detailed in this section to set up a connection.

Authentication plugin for connecting to Redshift using AWS IAM Identity Center

You can use AWS IAM Identity Center to connect to Amazon Redshift using the following driver plugins:

- `BrowserIdcAuthPlugin` – This plugin facilitates seamless single-sign-on integration with AWS IAM Identity Center. It creates a browser window for users to sign in with the user credentials defined in their corporate identity providers.
- `IdpTokenAuthPlugin` – This plugin should be used by applications that want to manage the authentication flow on their own, instead of letting the Amazon Redshift driver open a browser window for AWS IAM Identity Center authentication. It accepts an AWS IAM Identity Center vended Access token or an OpenID Connect (OIDC) JSON web token (JWT) from any web identity provider that's connected with AWS IAM Identity Center, such as Okta, PingOne, and Microsoft Entra ID (Azure AD). The client application is responsible for generating this required access token/JWT.

Authenticating with `BrowserIdcAuthPlugin`

Use the following plugin names to connect using `BrowserIdcAuthPlugin`, depending on your Amazon Redshift driver.

Driver	Connection option key	Value	Notes
JDBC	`plugin_name`	com.amazon.redshift.plugin.BrowserIdcAuthPlugin	You must enter the fully-qualified class name of the plugin when you connect.
ODBC	`plugin_name`	BrowserIdcAuthPlugin	
Python	`credentials_provider`	BrowserIdcAuthPlugin	There is no `plugin_name` option available for the Python driver. Instead, use `credentials_provider` .

The `BrowserIdcAuthPlugin` plugin has the following additional connection options:

Option name	Required?	Description	Example
idc_region	Required	The AWS Region where the AWS IAM Identity Center instance is located.	us-east-1
issuer_url	Required	The AWS IAM Identity Center server's instance endpoint. You can find this value using the AWS IAM Identity Center console.	https://identitycenter.amazonaws.com/ssoins-g5j2k70sn4yc5nsc

Option name	Required?	Description	Example
listen_port	Optional	The port that the Amazon Redshift driver uses to receive the `auth_code` response from AWS IAM Identity Center through the browser redirect.	7890
idc_client_display_name	Optional	The name that the AWS IAM Identity Center client uses for the application in the AWS IAM Identity Center's single sign-on consent popup.	Amazon Redshift driver
idp_response_timeout	Optional	The amount of time, in seconds, that the Redshift driver waits for the auth flow to complete.	60

You must enter these values in the connection properties of the tool you create and connect with. For more information, see the connection options documentation for each respective driver:

- Options for JDBC driver version 2.x configuration
- ODBC driver options
- Configuration options for the Amazon Redshift Python connector

Authenticating with `IdpTokenAuthPlugin`

Use the following plugin names to connect using `IdpTokenAuthPlugin`, depending on your Amazon Redshift driver.

Driver	Connection option key	Value	Notes
JDBC	`plugin_name`	com.amazon.redshift.plugin.IdpTokenAuthPlugin	You must enter the fully-qualified class name of the plugin when you connect.
ODBC	`plugin_name`	IdpTokenAuthPlugin	
Python	`credentials_provider`	IdpTokenAuthPlugin	There is no `plugin_name` option available for the Python driver. Instead, use `credentials_provider` .

The `IdpTokenAuthPlugin` plugin has the following additional connection options:

Option name	Required?	Description
token	Required	An AWS IAM Identity Center vended access token or an OpenID Connect (OIDC) JSON Web Token (JWT) provided by a web identity provider that's connected with AWS IAM Identity Center. Your application must generate this token by authenticating your application user with AWS IAM Identity Center or an identity provider connected with AWS IAM Identity Center.

Option name	Required?	Description
token_type	Required	The type of token used for `IdpTokenAuthPlugin` . Possible values are the following: • **ACCESS_TOKEN** – Enter this if you use an AWS IAM Identity Center provided access token. • **EXT_JWT** – Enter this if you use an OpenID Connect (OIDC) JSON Web Token (JWT) provided by a web-based identity provider that's connected with AWS IAM Identity Center.

You must enter these values in the connection properties of the tool you create and connect with. For more information, see the connection options documentation for each respective driver:

- Options for JDBC driver version 2.x configuration
- ODBC driver options
- Configuration options for the Amazon Redshift Python connector

Troubleshooting connections from Amazon Redshift query editor v2

This list details errors that commonly occur and can help you to connect to your Redshift database with query editor v2, using an AWS IAM Identity Center identity.

- Error: **Connection Issue: No Identity center session information available.** – When this error occurs, check your browser's security and privacy settings. These browser settings, particularly those for secure cookies, such as Firefox's Total Cookie Protection feature, can result in blocked connection attempts from Amazon Redshift query editor v2 to a Redshift database. Follow the remediation steps detailed for your browser:

- **Firefox** – Currently, third-party cookies are blocked by default. Click the shield in the browser's address bar and switch the toggle to turn off enhanced tracking protection for query editor v2.
- **Chrome incognito mode** – By default, Chrome Incognito mode blocks third party cookies. Click the eye icon in the address bar to allow third-party cookies for query editor v2. After you change the setting to allow cookies, you may not see the eye icon on the address bar.
- **Safari** – On a Mac, open the Safari app. Choose **Settings**, then choose **Advanced**. Toggle to turn off: **Block all cookies**.
- **Edge** – Choose **Settings**, then choose **Cookies and site permissions**. Then select **Manage and delete cookies and site data** and turn off **Block third-party cookies**.

  If you try to connect after changing the settings and continue to receive the error message **Connection Issue: No Identity center session information available**, we recommend that you refresh your connection with AWS IAM Identity Center. To do this, right click your Redshift database instance and choose **Refresh**. A new window appears, which you can use to authenticate.

- Error: **Connection issue: Identity center session expired or invalid.** – Following integration of a Redshift provisioned cluster or Serverless workgroup with AWS IAM Identity Center, a user might receive this error when they attempt to connect to a Redshift database from query editor v2. This can follow successful connection attempts. In this case, we recommend that you re-authenticate. To do this, right click your Redshift database instance and choose **Refresh**. A new window appears, which you can use to authenticate.
- Error: **Invalid scope. User credentials are not authorized to connect to Redshift.** – Following integration of a Redshift provisioned cluster or Serverless workgroup with AWS IAM Identity Center for identity management, a user might receive this error when they attempt to connect to a Redshift database from query editor v2. In this case, in order for query editor v2 to successfully connect and authenticate a user via AWS IAM Identity Center to access the correct resources, an administrator must assign the user to the Redshift AWS IAM Identity Center application through the Redshift console. This is completed under **IAM Identity Center connections**. Following this, the user can establish a successful connection after one hour, which is the limit of AWS IAM Identity Center session caching.
- Error: **Databases couldn't be listed. FATAL: Failed query when cluster is auto paused.** – When an Amazon Redshift Serverless database is in an idle state, not processing any workloads, it can remain paused when you connect with an AWS IAM Identity Center identity. To remedy this, log in with another authentication method to resume the Serverless workgroup. Then connect to the database with your AWS IAM Identity Center identity.

- Error: **An error occurred during the attempt to federate with AWS IAM Identity Center. An Amazon Redshift administrator must delete and recreate the AWS IAM Identity Center QEV2 application, using the Redshift console.** – This error typically occurs when the AWS IAM Identity Center applicaiton instance associated with query editor v2 is deleted. To remedy this, an Amazon Redshift administrator must delete and recreate the Redshift and query editor v2 applications for AWS IAM Identity Center. This can be performed on the Redshift console or using the https://docs.aws.amazon.com/cli/latest/reference/redshift/delete-redshift-idc-application.html CLI command.

Using service-linked roles for Amazon Redshift

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 add the necessary permissions manually. 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 or a Redshift-managed VPC endpoint. You can delete the service-linked role only after you delete all of the Amazon Redshift clusters or Redshift-managed VPC endpoints in your account. This protects your Amazon Redshift resources because you can't inadvertently remove permissions needed for access to the resources.

Amazon Redshift supports using service-linked roles in all of the Regions where the service is available. For more information, see AWS Regions and Endpoints.

For information about other services that support service-linked roles, see AWS services that work with IAM and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

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. This service-linked role is attached to the following managed policy: `AmazonRedshiftServiceLinkedRolePolicy`. For updates to this policy, see AWS-managed (predefined) policies for Amazon Redshift.

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`
- `ec2:CreateVpcEndpoint`
- `ec2:DeleteVpcEndpoints`
- `ec2:DescribeVpcEndpoints`
- `ec2:ModifyVpcEndpoint`
- `ec2:DescribeVpcAttribute`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeInternetGateways`
- `ec2:DescribeSecurityGroupRules`
- `ec2:DescribeAvailabilityZones`
- `ec2:DescribeNetworkAcls`
- `ec2:DescribeRouteTables`
- `ec2:AssignIpv6Addresses`
- `ec2:UnassignIpv6Addresses`

Permissions for network resources

The following permissions allow action on Amazon EC2 for creation and management of security group rules. These security groups and rules are specifically associated with the Amazon Redshift

`aws:RequestTag/Redshift` resource tag. This limits the scope of the permissions to specific Amazon Redshift resources.

- `ec2:CreateSecurityGroup`
- `ec2:AuthorizeSecurityGroupEgress`
- `ec2:AuthorizeSecurityGroupIngress`
- `ec2:RevokeSecurityGroupEgress`
- `ec2:RevokeSecurityGroupIngress`
- `ec2:ModifySecurityGroupRules`
- `ec2:DeleteSecurityGroup`

Permissions for service quotas

The following permissions allow the caller to get service quotas.

`servicequotas:GetServiceQuota`

The following JSON fragment shows action and resource scope for service quotas.

```
{
   "Sid": "ServiceQuotasToCheckCustomerLimits",
   "Effect": "Allow",
   "Action": [
      "servicequotas:GetServiceQuota"
   ],
   "Resource": [
      "arn:aws:servicequotas:*:*:ec2/L-0263D0A3",
      "arn:aws:servicequotas:*:*:vpc/L-29B6F2EB"
   ]
}
```

The quota codes are the following:

- *L-0263D0A3* – The quota code for EC2-VPC Elastic IPs.
- *L-29B6F2EB* – The quota code for Interface VPC endpoints per VPC.

For more information, see AWS service quotas.

Actions for audit logging

Actions listed with the `logs` prefix pertain to audit logging and related features. Specifically, creation and management of log groups and log streams.

- `logs:CreateLogGroup`
- `logs:PutRetentionPolicy`
- `logs:CreateLogStream`
- `logs:PutLogEvents`
- `logs:DescribeLogStreams`
- `logs:GetLogEvents`

The following JSON shows actions and resource scope, to Amazon Redshift, for audit logging.

```
[
    {
        "Sid": "EnableCreationAndManagementOfRedshiftCloudwatchLogGroups",
        "Effect": "Allow",
        "Action": [
            "logs:CreateLogGroup",
            "logs:PutRetentionPolicy"
        ],
        "Resource": [
            "arn:aws:logs:*:*:log-group:/aws/redshift/*"
        ]
    },
    {
        "Sid": "EnableCreationAndManagementOfRedshiftCloudwatchLogStreams",
        "Effect": "Allow",
        "Action": [
            "logs:CreateLogStream",
            "logs:PutLogEvents",
            "logs:DescribeLogStreams",
            "logs:GetLogEvents"
        ],
        "Resource": [
            "arn:aws:logs:*:*:log-group:/aws/redshift/*:log-stream:*"
        ]
    }
]
```

For more information about service-linked roles and their purpose in AWS, see Using service-linked roles. For more information about specific actions and other IAM resources for Amazon Redshift, see Actions, resources, and condition keys for Amazon Redshift.

Actions for managing admin credentials with AWS Secrets Manager

Actions listed with the `secretsmanager` prefix pertain to using Amazon Redshift to manage your admin credentials. These actions let Amazon Redshift use AWS Secrets Manager to create and manage your admin credential secrets.

The following JSON shows actions and resource scope, to Amazon Redshift, for managing admin credentials with AWS Secrets Manager.

```
[
    {
        "Effect": "Allow",
        "Action": [
            "secretsmanager:DescribeSecret",
            "secretsmanager:DeleteSecret",
            "secretsmanager:PutSecretValue",
            "secretsmanager:UpdateSecret",
            "secretsmanager:UpdateSecretVersionStage",
            "secretsmanager:RotateSecret"
        ],
        "Resource": [
            "arn:aws:secretsmanager:*:*:secret:redshift!*"
        ],
        "Condition": {
            "StringEquals": {
                "secretsmanager:ResourceTag/aws:secretsmanager:owningService":
 "redshift"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "secretsmanager:GetRandomPassword"
        ],
        "Resource": "*"
    }
]
```

Actions for registering clusters and serverless namespaces to the AWS Glue Data Catalog

Actions listed with the `glue` prefix pertain to accessing catalogs in the AWS Glue Data Catalog made from registering provisioned clusters or serverless namespaces. For more information, see Apache Iceberg compatibility for Amazon Redshift in the *Amazon Redshift Database Developer Guide*.

The following JSON shows actions and resource scope, to Amazon Redshift, for accessing catalogs in the AWS Glue Data Catalog:

```
[
    {
        "Sid": "DiscoverRedshiftCatalogs",
        "Effect": "Allow",
        "Action": [
            "glue:GetCatalogs",
            "glue:GetCatalog"
        ],
        "Resource": [
            "arn:aws:glue:*:*:catalog",
            "arn:aws:glue:*:*:catalog/*"
        ],
   "Condition":
    {
        "Bool":
        {
            "glue:EnabledForRedshiftAutoDiscovery": "true"
        },
        "StringEquals": {
             "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
    }
 },
 {
    "Sid": "LakeFormationGetMetadataAccessForFederatedCatalogs",
    "Effect": "Allow",
    "Action": [
        "lakeformation:GetDataAccess"
    ],
    "Resource": [ "*" ],
    "Condition":
    {
        "Bool":
```

```
        {
            "lakeformation:EnabledOnlyForMetaDataAccess":"true"
        },
        "StringEquals": {
             "aws:ResourceAccount": "${aws:PrincipalAccount}"
        },
        "ForAnyValue:StringEquals":
        {
            "aws:CalledVia": "glue.amazonaws.com"
        }
    }
 }
    }
]
```

The `glue:GetCatalog` and `glue:GetCatalogs` permissions have the condition `glue:EnabledForRedshiftAutoDiscovery:true`, which means that Amazon Redshift grants IAM access for automatically discovering catalogs. To opt-out, add an AWS Glue account-level resource policy to selectively deny service-linked role access to the catalogs. Since the service-linked role already has an explicit allow action in the policy, the opt-out policy needs to explicitly deny that action. Consider the following example, where an additional policy denies auto discovery for Amazon Redshift:

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : {
            "Effect": "Deny",
            "Action": [
                "glue:GetCatalog",
                "glue:GetCatalogs"
            ],
            "Principal" : {
            "AWS" : "arn:aws:iam::111122223333:role/aws-service-role/
redshift.amazonaws.com/AWSServiceRoleForRedshift"
            },
            "Resource": [
                "arn:aws:glue:*:*:catalog/<s3_table_catalog_name>",
                "arn:aws:glue:*:*:catalog/<s3_table_catalog_name>/*"
            ]
```

```
        }
}
```

To allow an IAM entity to create AWSServiceRoleForRedshift service-linked roles

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole"
    ],
    "Resource": "arn:aws:iam::<AWS-account-ID>:role/aws-service-role/
redshift.amazonaws.com/AWSServiceRoleForRedshift",
    "Condition": {"StringLike": {"iam:AWSServiceName": "redshift.amazonaws.com"}}
}
```

To allow an IAM entity to delete AWSServiceRoleForRedshift service-linked roles

Add the following policy statement to the permissions for that IAM entity:

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::<AWS-account-ID>:role/aws-service-role/
redshift.amazonaws.com/AWSServiceRoleForRedshift",
    "Condition": {"StringLike": {"iam:AWSServiceName": "redshift.amazonaws.com"}}
}
```

Alternatively, you can use an AWS managed policy to provide full access to Amazon Redshift.

Creating a service-linked role for Amazon Redshift

You don't need to manually create an AWSServiceRoleForRedshift service-linked role. Amazon Redshift creates the service-linked role for you. If the AWSServiceRoleForRedshift service-linked role has been deleted from your account, Amazon Redshift creates the role when you launch a new Amazon Redshift cluster.

Important

If you used the Amazon Redshift service before September 18, 2017, when it began supporting service-linked roles, then Amazon Redshift created the AWSServiceRoleForRedshift role in your account. To learn more, see A new role appeared in my IAM account.

Editing a service-linked role for Amazon Redshift

Amazon Redshift does not allow you to edit the AWSServiceRoleForRedshift service-linked role. After you create a service-linked role, you can't change the name of the role because various entities might reference the role. However, you can edit the description of the role using the IAM console, the AWS Command Line Interface (AWS CLI), or IAM API. For more information, see Modifying a role in the *IAM User Guide*.

Deleting a service-linked role for Amazon Redshift

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained.

Before you can delete a service-linked role for an account, you must shut down and delete any clusters in the account. For more information, see Shutting down and deleting a cluster.

You can use the IAM console, the AWS CLI, or the IAM API to delete a service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Using IAM authentication to generate database user credentials

You can generate temporary database credentials based on permissions granted through an AWS Identity and Access Management (IAM) permissions policy to manage the access that your users have to your Amazon Redshift database.

Commonly, Amazon Redshift database users log in to the database by providing a database user name and password. However, you don't have to maintain user names and passwords in your Amazon Redshift database. As an alternative, you can configure your system to permit users to create user credentials and log in to the database based on their IAM credentials.

Amazon Redshift provides the GetClusterCredentials API operation to generate temporary database user credentials. You can configure your SQL client with Amazon Redshift JDBC or ODBC drivers that manage the process of calling the `GetClusterCredentials` operation. They do so by retrieving the database user credentials, and establishing a connection between your SQL client and your Amazon Redshift database. You can also use your database application to programmatically call the `GetClusterCredentials` operation, retrieve database user credentials, and connect to the database.

If you already manage user identities outside AWS, you can use an identity provider (IdP) compliant with Security Assertion Markup Language (SAML) 2.0 to manage access to Amazon Redshift resources. You configure your IdP to permit your federated users access to an IAM role. With that IAM role, you can generate temporary database credentials and log in to Amazon Redshift databases.

Your SQL client needs permission to call the `GetClusterCredentials` operation for you. You manage those permissions by creating an IAM role and attaching an IAM permissions policy that grants or restricts access to the `GetClusterCredentials` operation and related actions. As a best practice, we recommend attaching permissions policies to an IAM role and then assigning it to users and groups as needed. For more information, see Identity and access management in Amazon Redshift.

The policy also grants or restricts access to specific resources, such as Amazon Redshift clusters, databases, database user names, and user group names.

Note

We recommend using the Amazon Redshift JDBC or ODBC drivers to manage the process of calling the `GetClusterCredentials` operation and logging on to the database. For simplicity, we assume that you are using a SQL client with the JDBC or ODBC drivers throughout this topic.
For specific details and examples of using the `GetClusterCredentials` operation or the parallel `get-cluster-credentials` CLI command, see GetClusterCredentials and get-cluster-credentials.

To manage authentication and authorization centrally, Amazon Redshift supports database authentication with IAM, enabling user authentication through enterprise federation. Instead of creating a user, you can use existing identities from AWS Directory Service, your enterprise user

directory, or a web identity provider. These are known as federated users. AWS assigns a role to a federated user when access is requested through an IdP.

To provide federated access to a user or client application in your organization to call Amazon Redshift API operations, you can also use the JDBC or ODBC driver with SAML 2.0 support to request authentication from your organization IdP. In this case, your organization's users don't have direct access to Amazon Redshift.

For more information, see Identity Providers and Federation in the *IAM User Guide*.

Creating temporary IAM credentials

In this section, you can find how to configure your system to generate temporary IAM-based database user credentials and log in to your database using the new credentials.

At a high level, the process flows as follows:

1. Step 1: Create an IAM role for IAM single sign-on access

   (Optional) You can authenticate users for access to an Amazon Redshift database by integrating IAM authentication and a third-party identity provider (IdP).
2. Step 2: Configure SAML assertions for your IdP

   (Optional) To use IAM authentication using an IdP, you need to define a claim rule in your IdP application that maps users or groups in your organization to the IAM role. Optionally, you can include attribute elements to set `GetClusterCredentials` parameters.
3. Step 3: Create an IAM role with permissions to call GetClusterCredentialsWithIAM or GetClusterCredentials

   Your SQL client application assumes the user when it calls the `GetClusterCredentials` operation. If you created an IAM role for identity provider access, you can add the necessary permission to that role.
4. Step 4: Create a database user and database groups

   (Optional) By default, `GetClusterCredentials` returns credentials create a new user if the user name doesn't exist. You can also choose to specify user groups that users join at logon. By default, database users join the PUBLIC group.
5. Step 5: Configure a JDBC or ODBC connection to use IAM credentials

To connect to your Amazon Redshift database, you configure your SQL client to use an Amazon Redshift JDBC or ODBC driver.

Step 1: Create an IAM role for IAM single sign-on access

If you don't use an identity provider for single sign-on access, you can skip this step.

If you already manage user identities outside of AWS, you can authenticate users for access to an Amazon Redshift database by integrating IAM authentication and a third-party SAML-2.0 identity provider (IdP).

For more information, see Identity Providers and Federation in the *IAM User Guide*.

Before you can use Amazon Redshift IdP authentication, create an AWS SAML identity provider. You create an IdP in the IAM console to inform AWS about the IdP and its configuration. Doing this establishes trust between your AWS account and the IdP. For steps to create a role, see Creating a Role for SAML 2.0 Federation (Console) in the *IAM User Guide*.

Step 2: Configure SAML assertions for your IdP

After you create the IAM role, you define a claim rule in your IdP application to map users or groups in your organization to the IAM role. For more information, see Configuring SAML Assertions for the Authentication Response in the *IAM User Guide*.

If you choose to use the optional `GetClusterCredentials` parameters `DbUser`, `AutoCreate`, and `DbGroups`, you have two options. You can set the values for the parameters with your JDBC or ODBC connection, or you can set the values by adding SAML attribute elements to your IdP. For more information about the `DbUser`, `AutoCreate`, and `DbGroups` parameters, see Step 5: Configure a JDBC or ODBC connection to use IAM credentials.

Note

If you use an IAM policy variable `${redshift:DbUser}`, as described in Resource policies for GetClusterCredentials the value for `DbUser` is replaced with the value retrieved by the API operation's request context. The Amazon Redshift drivers use the value for the `DbUser` variable provided by the connection URL, rather than the value supplied as a SAML attribute.
To help secure this configuration, we recommend that you use a condition in an IAM policy to validate the `DbUser` value by using `RoleSessionName`. You can find examples

of how to set a condition using an IAM policy in Example 8: IAM policy for using GetClusterCredentials.

To configure your IdP to set the DbUser, AutoCreate, and DbGroups parameters, include the following Attribute elements:

- An Attribute element with the Name attribute set to "https://redshift.amazon.com/SAML/Attributes/DbUser"

  Set the AttributeValue element to the name of a user that will connect to the Amazon Redshift database.

  The value in the AttributeValue element must be lowercase, begin with a letter, contain only alphanumeric characters, underscore ('_'), plus sign ('+'), dot ('.'), at ('@'), or hyphen ('-'), and be fewer than 128 characters. Typically, the user name is a user ID (for example, bobsmith) or an email address (for example bobsmith@example.com). The value can't include a space (for example, a user's display name such as Bob Smith).

```
<Attribute Name="https://redshift.amazon.com/SAML/Attributes/DbUser">
    <AttributeValue>user-name</AttributeValue>
</Attribute>
```

- An Attribute element with the Name attribute set to "https://redshift.amazon.com/SAML/Attributes/AutoCreate"

  Set the AttributeValue element to true to create a new database user if one doesn't exist. Set the AttributeValue to false to specify that the database user must exist in the Amazon Redshift database.

```
<Attribute Name="https://redshift.amazon.com/SAML/Attributes/AutoCreate">
    <AttributeValue>true</AttributeValue>
</Attribute>
```

- An Attribute element with the Name attribute set to set to "https://redshift.amazon.com/SAML/Attributes/DbGroups"

  This element contains one or more AttributeValue elements. Set each AttributeValue element to a database group name that the DbUser joins for the duration of the session when connecting to the Amazon Redshift database.

```
<Attribute Name="https://redshift.amazon.com/SAML/Attributes/DbGroups">
    <AttributeValue>group1</AttributeValue>
    <AttributeValue>group2</AttributeValue>
    <AttributeValue>group3</AttributeValue>
</Attribute>
```

Step 3: Create an IAM role with permissions to call GetClusterCredentialsWithIAM or GetClusterCredentials

Your SQL client needs authorization to call the the `GetClusterCredentialsWithIAM` or `GetClusterCredentials` operation on your behalf. To provide that authorization, you create a user or role and attach a policy that grants the necessary permissions. Both operations are available for obtaining the cluster credentials, but they differ in their authentication method. `GetClusterCredentialsWithIAM` uses an IAM role, automatically creating a database user that maps to the role, which is beneficial for managing permissions at the IAM role level, while `GetClusterCredentials` gives credentials for a given username in the database.

To create an IAM role with permissions to call GetClusterCredentialsWithIAM

1. Using the IAM service, create a user or role. You can also use an existing user or role. For example, if you created an IAM role for identity provider access, you can attach the necessary IAM policies to that role.
2. Attach a permission policy with permission to call the `redshift:GetClusterCredentialsWithIAM` operation. The following policy sample shows options to allow the operation for specific cluster and database, any database in cluster, and any database in any cluster.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SpecificClusterAndDBName",
            "Effect": "Allow",
            "Action": "redshift:GetClusterCredentialsWithIAM",
            "Resource": [
                "arn:aws:redshift:us-east-1:123456789012:dbname:testcluster/
testdatabase"
            ]
```

```
        },
        {
            "Sid": "SpecificClusterAndAnyDBName",
            "Effect": "Allow",
            "Action": "redshift:GetClusterCredentialsWithIAM",
            "Resource": "arn:aws:redshift:us-
east-1:123456789012:dbname:examplecluster/*",
        },
        {
            "Sid": "AnyClusterAnyDatabase",
            "Effect": "Allow",
            "Action": "redshift:GetClusterCredentialsWithIAM",
            "Resource": "*"
        }
    ]
}
```

To create an IAM role with permissions to call GetClusterCredentials

1. Using the IAM service, create a user or role. You can also use an existing user or role. For example, if you created an IAM role for identity provider access, you can attach the necessary IAM policies to that role.
2. Attach a permission policy with permission to call the `redshift:GetClusterCredentials` operation. Depending on which optional parameters you specify, you can also allow or restrict additional actions and resources in your policy:

   - To permit your SQL client to retrieve cluster ID, AWS Region, and port, include permission to call the `redshift:DescribeClusters` operation with the Redshift cluster resource.
   - If you use the `AutoCreate` option, include permission to call `redshift:CreateClusterUser` with the `dbuser` resource. The following Amazon Resource Name (ARN) specifies the Amazon Redshift `dbuser`. Replace *`region`*, *`account-id`*, and *`cluster-name`* with the values for your AWS Region, account, and cluster. For *`dbuser-name`*, specify the user name to use to log in to the cluster database.

     ```
     arn:aws:redshift:region:account-id:dbuser:cluster-name/dbuser-name
     ```

   - (Optional) Add an ARN that specifies the Amazon Redshift `dbname` resource in the following format. Replace *`region`*, *`account-id`*, and *`cluster-name`* with the values for your AWS

Region, account, and cluster. For *database-name*, specify the name of a database that the user will log in to.

```
arn:aws:redshift:region:account-id:dbname:cluster-name/database-name
```

- If you use the DbGroups option, include permission to call the `redshift:JoinGroup` operation with the Amazon Redshift dbgroup resource in the following format. Replace *region*, *account-id*, and *cluster-name* with the values for your AWS Region, account, and cluster. For *dbgroup-name*, specify the name of a user group that the user joins at login.

```
arn:aws:redshift:region:account-id:dbgroup:cluster-name/dbgroup-name
```

For more information and examples, see Resource policies for GetClusterCredentials.

Step 4: Create a database user and database groups

Optionally, you can create a database user that you use to log in to the cluster database. If you create temporary user credentials for an existing user, you can disable the user's password to force the user to log on with the temporary password. Alternatively, you can use the `GetClusterCredentials` Autocreate option to automatically create a new database user.

You can create database user groups with the permissions you want the IAM database user to join at login. When you call the `GetClusterCredentials` operation, you can specify a list of user group names that the new user joins at login. These group memberships are valid only for sessions created using credentials generated with the given request.

To create a database user and database groups

1. Log in to your Amazon Redshift database and create a database user using CREATE USER or alter an existing user using ALTER USER.
2. Optionally, specify the PASSWORD DISABLE option to prevent the user from using a password. When a user's password is disabled, the user can log on only using temporary credentials. If the password is not disabled, the user can log on either with the password or using temporary credentials. You can't disable the password for a superuser.

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
IAM	(Recommended) Use console credentials as temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Login for AWS local development in the *AWS Command Line Interface User Guide*. • For AWS SDKs, see Login for AWS local development in the *AWS SDKs and Tools Reference Guide*.

Which user needs programmatic access?	To	By
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.

Which user needs programmatic access?	To	By
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

The following example creates a user with password disabled.

```
create user temp_creds_user password disable;
```

The following example disables the password for an existing user.

```
alter user temp_creds_user password disable;
```

3. Create database user groups using CREATE GROUP.
4. Use the GRANT command to define access privileges for the groups.

Step 5: Configure a JDBC or ODBC connection to use IAM credentials

You can configure your SQL client with an Amazon Redshift JDBC or ODBC driver. This driver manages the process of creating database user credentials and establishing a connection between your SQL client and your Amazon Redshift database.

If you use an identity provider for authentication, specify the name of a credential provider plugin. The Amazon Redshift JDBC and ODBC drivers include plugins for the following SAML-based identity providers:

- Active Directory Federation Services (AD FS)
- PingOne
- Okta
- Microsoft Azure AD

  For the steps to set up Microsoft Azure AD as an identity provider, see Setting up JDBC or ODBC single sign-on authentication.

To configure a JDBC connection to use IAM credentials

1. Download the latest Amazon Redshift JDBC driver from the Configuring a connection for JDBC driver version 2.x for Amazon Redshift page.
2. Create a JDBC URL with the IAM credentials options in one of the following formats. To use IAM authentication, add `iam:` to the Amazon Redshift JDBC URL following `jdbc:redshift:` as shown in the following example.

   ```
   jdbc:redshift:iam://
   ```

   Add `cluster-name`, `region`, and `account-id`. The JDBC driver uses your IAM account information and cluster name to retrieve the cluster ID and AWS Region. To do so, your user or role must have permission to call the `redshift:DescribeClusters` operation with the specified cluster. If your user or role doesn't have permission to call the `redshift:DescribeClusters` operation, include the cluster ID, AWS Region, and port as shown in the following example. The port number is optional.

   ```
   jdbc:redshift:iam://examplecluster.abc123xyz789.us-
   west-2.redshift.amazonaws.com:5439/dev
   ```

3. Add JDBC options to provide IAM credentials. You use different combinations of JDBC options to provide IAM credentials. For details, see JDBC and ODBC options for creating database user credentials.

   The following URL specifies AccessKeyID and SecretAccessKey for a user.

   ```
   jdbc:redshift:iam://examplecluster:us-west-2/dev?
   AccessKeyID=AKIAIOSFODNN7EXAMPLE&SecretAccessKey=wJalrXUtnFEMI/K7MDENG/
   bPxRfiCYEXAMPLEKEY
   ```

   The following example specifies a named profile that contains the IAM credentials.

   ```
   jdbc:redshift:iam://examplecluster:us-west-2/dev?Profile=user2
   ```

4. Add JDBC options that the JDBC driver uses to call the `GetClusterCredentials` API operation. Don't include these options if you call the `GetClusterCredentials` API operation programmatically.

   The following example includes the JDBC `GetClusterCredentials` options.

   ```
   jdbc:redshift:iam://examplecluster:us-west-2/dev?
   plugin_name=com.amazon.redshift.plugin.AzureCredentialsProvider&UID=user&PWD=password&idp_t
   ```

To configure an ODBC connection to use IAM credentials

In the following procedure, you can find steps only to configure IAM authentication. For steps to use standard authentication, using a database user name and password, see Configuring a connection for ODBC driver version 2.x for Amazon Redshift.

1. Install and configure the latest Amazon Redshift OBDC driver for your operating system. For more information, see Configuring a connection for ODBC driver version 2.x for Amazon Redshift page.

   > **Important**
   >
   > The Amazon Redshift ODBC driver must be version 1.3.6.1000 or later.

2. Follow the steps for your operating system to configure connection settings.

3. On Microsoft Windows operating systems, access the Amazon Redshift ODBC Driver DSN Setup window.

   a. Under **Connection Settings**, enter the following information:

      - **Data Source Name**
      - **Server** (optional)
      - **Port** (optional)
      - **Database**

      If your user or role has permission to call the `redshift:DescribeClusters` operation, only **Data Source Name** and **Database** are required. Amazon Redshift uses **ClusterId** and **Region** to get the server and port by calling the `DescribeCluster` operation.

      If your user or role doesn't have permission to call the `redshift:DescribeClusters` operation, specify **Server** and **Port**.

   b. Under **Authentication**, choose a value for **Auth Type**.

      For each authentication type, enter values as listed following:

      AWS Profile

      Enter the following information:

      - **ClusterID**
      - **Region**
      - **Profile name**

        Enter the name of a profile in an AWS config file that contains values for the ODBC connection options. For more information, see Using a configuration profile.

      (Optional) Provide details for options that the ODBC driver uses to call the `GetClusterCredentials` API operation:

      - **DbUser**
      - **User AutoCreate**
      - **DbGroups**

For more information, see JDBC and ODBC options for creating database user credentials.

IAM Credentials

Enter the following information:

- **ClusterID**
- **Region**
- **AccessKeyID** and **SecretAccessKey**

  The access key ID and secret access key for the IAM role or user configured for IAM database authentication.

- **SessionToken**

  SessionToken is required for an IAM role with temporary credentials. For more information, see Temporary Security Credentials.

Provide details for options that the ODBC driver uses to call the `GetClusterCredentials` API operation:

- **DbUser** (required)
- **User AutoCreate** (optional)
- **DbGroups** (optional)

  For more information, see JDBC and ODBC options for creating database user credentials.

Identity Provider: AD FS

For Windows Integrated Authentication with AD FS, leave **User** and **Password** empty.

Provide IdP details:

- **IdP Host**

  The name of the corporate identity provider host. This name should not include any slashes (/).

- **IdP Port** (optional)

The port used by identity provider. The default is 443.

- **Preferred Role**

  An Amazon Resource Name (ARN) for the IAM role from the multi-valued `AttributeValue` elements for the `Role` attribute in the SAML assertion. To find the appropriate value for the preferred role, work with your IdP administrator. For more information, see Step 2: Configure SAML assertions for your IdP.

(Optional) Provide details for options that the ODBC driver uses to call the `GetClusterCredentials` API operation:

- **DbUser**
- **User AutoCreate**
- **DbGroups**

For more information, see JDBC and ODBC options for creating database user credentials.

Identity Provider: PingFederate

For **User** and **Password**, enter your IdP user name and password.

Provide IdP details:

- **IdP Host**

  The name of the corporate identity provider host. This name should not include any slashes (/).

- **IdP Port** (optional)

  The port used by identity provider. The default is 443.

- **Preferred Role**

  An Amazon Resource Name (ARN) for the IAM role from the multi-valued `AttributeValue` elements for the `Role` attribute in the SAML assertion. To find the appropriate value for the preferred role, work with your IdP administrator. For more information, see Step 2: Configure SAML assertions for your IdP.

(Optional) Provide details for options that the ODBC driver uses to call the `GetClusterCredentials` API operation:

- **DbUser**
- **User AutoCreate**
- **DbGroups**

For more information, see JDBC and ODBC options for creating database user credentials.

Identity Provider: Okta

For **User** and **Password**, enter your IdP user name and password.

Provide IdP details:

- **IdP Host**

  The name of the corporate identity provider host. This name should not include any slashes (/).

- **IdP Port**

  This value is not used by Okta.

- **Preferred Role**

  An Amazon Resource Name (ARN) for the IAM role from the `AttributeValue` elements for the `Role` attribute in the SAML assertion. To find the appropriate value for the preferred role, work with your IdP administrator. For more information, see Step 2: Configure SAML assertions for your IdP.

- **Okta App ID**

  An ID for an Okta application. The value for App ID follows "amazon_aws" in the Okta application embed link. Work with your IdP administrator to get this value.

(Optional) Provide details for options that the ODBC driver uses to call the `GetClusterCredentials` API operation:

- **DbUser**
- **User AutoCreate**

- **DbGroups**

For more information, see JDBC and ODBC options for creating database user credentials.

Identity Provider: Azure AD

For **User** and **Password**, enter your IdP user name and password.

For **Cluster ID** and **Region**, enter the cluster ID and AWS Region of your Amazon Redshift cluster.

For **Database**, enter the database that you created for your Amazon Redshift cluster.

Provide IdP details:

- **IdP Tenant**

  The tenant used for Azure AD.

- **Azure Client Secret**

  The client secret of the Amazon Redshift enterprise app in Azure.

- **Azure Client ID**

  The client ID (application ID) of the Amazon Redshift enterprise app in Azure.

(Optional) Provide details for options that the ODBC driver uses to call the `GetClusterCredentials` API operation:

- **DbUser**
- **User AutoCreate**
- **DbGroups**

For more information, see JDBC and ODBC options for creating database user credentials.

Options for providing IAM credentials

To provide IAM credentials for a JDBC or ODBC connection, choose one of the following options.

- **AWS profile**

  As an alternative to providing credentials values in the form of JDBC or ODBC settings, you can put the values in a named profile. For more information, see Using a configuration profile.

- **IAM credentials**

  Provide values for AccessKeyID, SecretAccessKey, and, optionally, SessionToken in the form of JDBC or ODBC settings. SessionToken is required only for an IAM role with temporary credentials. For more information, see JDBC and ODBC options for providing IAM credentials.

- **Identity provider federation**

  When you use identity provider federation to enable users from an identity provider to authenticate to Amazon Redshift, specify the name of a credential provider plugin. For more information, see Credentials provider plugins.

  The Amazon Redshift JDBC and ODBC drivers include plugins for the following SAML-based identity federation credential providers:

  - Microsoft Active Identity Federation Services (AD FS)
  - PingOne
  - Okta
  - Microsoft Azure Active Directory (Azure AD)

  You can provide the plugin name and related values in the form of JDBC or ODBC settings or by using a profile. For more information, see Options for JDBC driver version 2.x configuration.

For more information, see Step 5: Configure a JDBC or ODBC connection to use IAM credentials.

Using a configuration profile

You can supply the IAM credentials options and `GetClusterCredentials` options as settings in named profiles in your AWS configuration file. To provide the profile name, use the Profile JDBC option. The configuration is stored in a file named `config` or a file named `credentials` in a folder named `.aws` in your home directory.

For a SAML-based credential provider plugin included with an Amazon Redshift JDBC or ODBC driver, you can use the settings described just preceding in Credentials provider plugins. If `plugin_name` isn't used, the other options are ignored.

The following example shows the ~/.aws/credentials file with two profiles.

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

[user2]
aws_access_key_id=AKIAI44QH8DHBEXAMPLE
aws_secret_access_key=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
session_token=AQoDYXdzEPT//////////
wEXAMPLEtc764bNrC9SAPBSM22wDOk4x4HIZ8j4FZTwdQWLWsKWHGBuFqwAeMicRXmxfpSPfIeoIYRqTflfKD8YUuwthAx7
qkPpKPi/kMcGd
QrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDy0KPkyQDYwT7WZ0wq5VSXDvp75YU
9HFvlRd8Tx6q6fE8YQcHNVXAkiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL64lIZbqBAz
+scqKmlzm8FDrypNC9Yjc8fPOLn9FX9KSYvKTr4rvx3iSIlTJabIQwj2ICCR/oLxBA==
```

To use the credentials for the `user2` example, specify `Profile=user2` in the JDBC URL.

For more information on using profiles, see Configuration and credential file settings in the *AWS Command Line Interface User Guide.*

For more information on using profiles for the JDBC driver, see Specifying profiles.

For more information on using profiles for the ODBC driver, see Authentication methods.

JDBC and ODBC options for providing IAM credentials

The following table lists the JDBC and ODBC options for providing IAM credentials.

Option	Description
`Iam`	For use only in an ODBC connection string. Set to 1 to use IAM authentication.

`AccessKeyID`
 	
`SecretAccessKey`
 	
`SessionToken`
	The access key ID and secret access key for the IAM role or user configured for IAM database authentication. `SessionToken` is required only for an IAM role with temporary credentials. SessionToken isn't used for a user. For more information, see Temporary Security Credentials.

Option	Description
`plugin_name`	The fully qualified name of a class that implements a credentials provider. The Amazon Redshift JDBC driver includes SAML-based credential provider plugins. If you provide `plugin_name` , you can also provide other related options. For more information, see Credentials provider plugins.
`Profile`	The name of a profile in an AWS credentials or config file that contains values for the JDBC connection options. For more information, see Using a configuration profile.

JDBC and ODBC options for creating database user credentials

To use the Amazon Redshift JDBC or ODBC driver to create database user credentials, provide the database user name as a JDBC or ODBC option. Optionally, you can have the driver create a new database user if one doesn't exist, and you can specify a list of database user groups the user joins at login.

If you use an identity provider (IdP), work with your IdP administrator to determine the correct values for these options. Your IdP administrator can also configure your IdP to provide these options, in which case you don't need to provide them as JDBC or ODBC options. For more information, see Step 2: Configure SAML assertions for your IdP.

Note

If you use an IAM policy variable `${redshift:DbUser}`, as described in Resource policies for GetClusterCredentials the value for `DbUser` is replaced with the value retrieved by the API operation's request context. The Amazon Redshift drivers use the value for the `DbUser` variable provided by the connection URL, rather than the value supplied as a SAML attribute.
To help secure this configuration, we recommend that you use a condition in an IAM policy to validate the `DbUser` value with the `RoleSessionName`. You can find examples of how to set a condition using an IAM policy in Example 8: IAM policy for using GetClusterCredentials.

The following table lists the options for creating database user credentials.

Option	Description
DbUser	The name of a database user. If a user named DbUser exists in the database, the temporary user credentials have the same permissions as the existing user. If DbUser doesn't exist in the database and AutoCreate is true, a new user named DbUser is created. Optionally, disable the password for an existing user. For more information, see ALTER_USER
AutoCreate	Specify `true` to create a database user with the name specified for DbUser if one does not exist. The default is false.
`DbGroups`	A comma-delimited list of the names of one or more existing database groups the database user joins for the current session. By default, the new user is added only to PUBLIC.

Credentials provider plugins

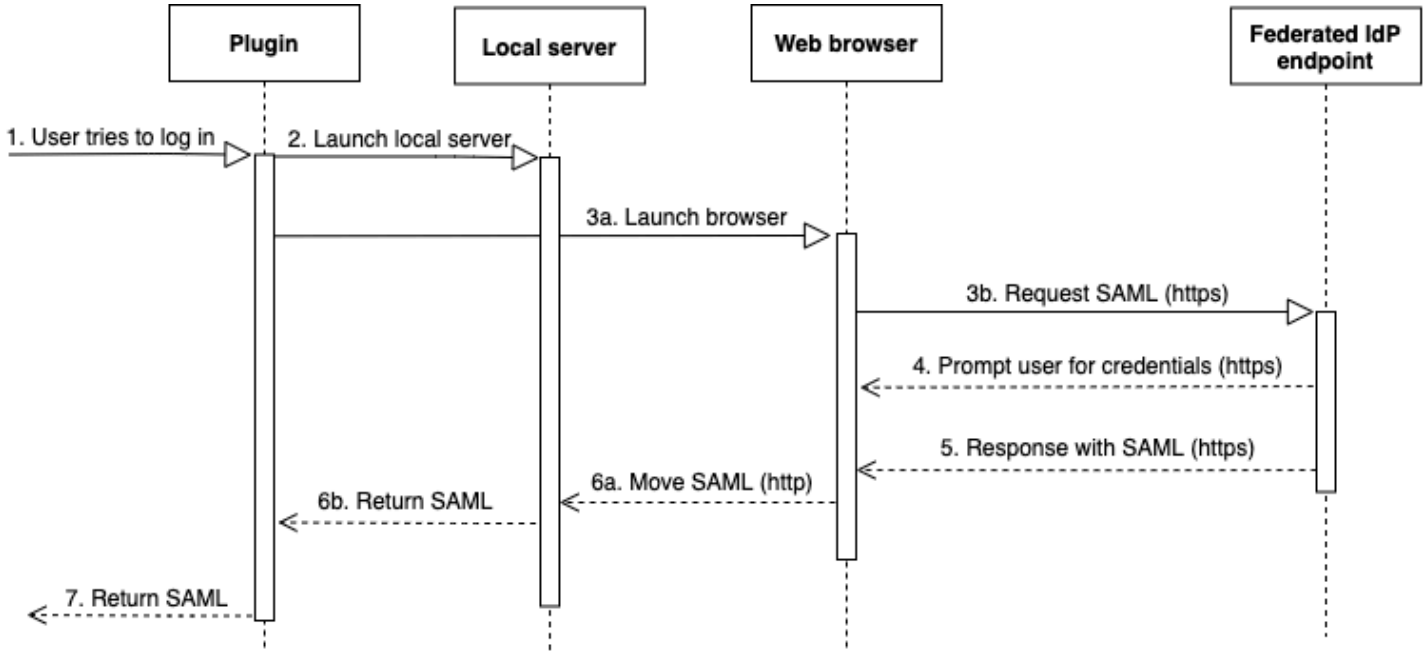
Amazon Redshift uses credentials provider plugins for single sign-on authentication.

To support single sign-on authentication, Amazon Redshift provides the Azure AD plugin for Microsoft Azure Active Directory. For information on how to configure this plugin, see Setting up JDBC or ODBC single sign-on authentication.

Multi-factor authentication
Multi-factor authentication

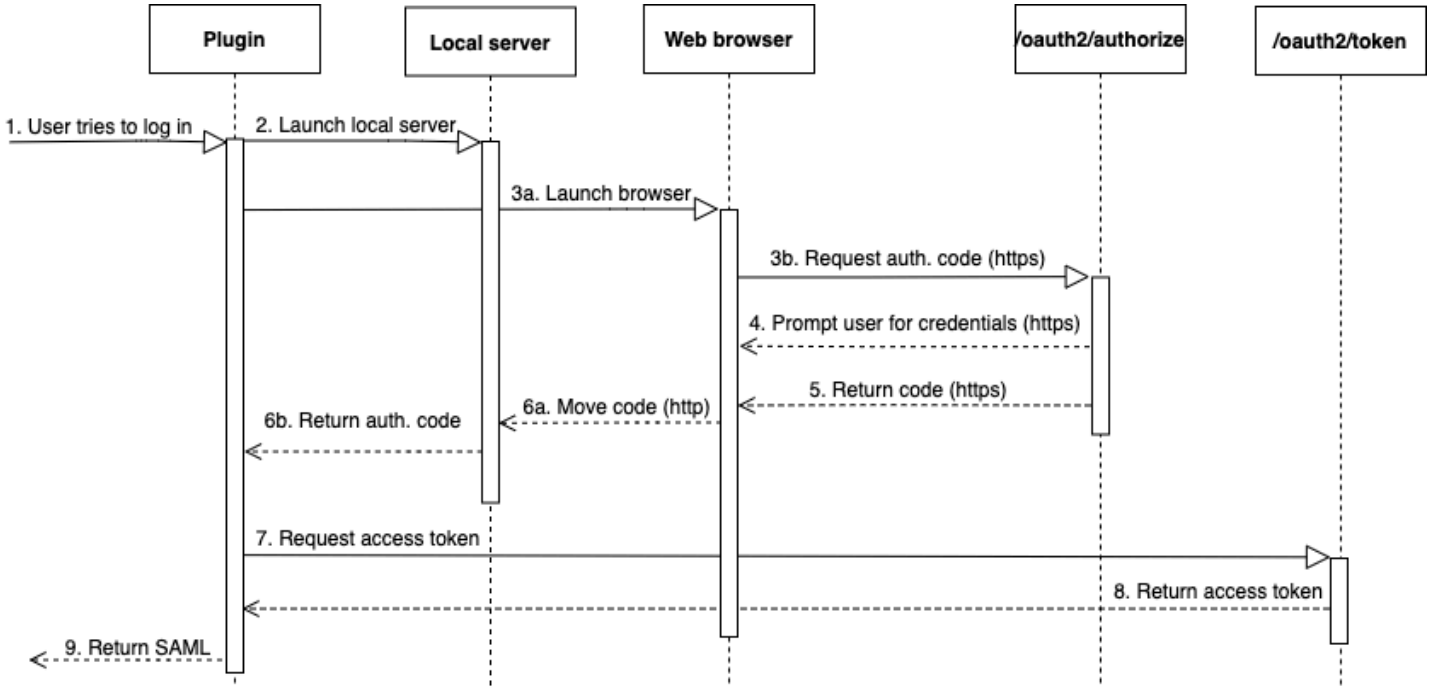
To support multi-factor authentication (MFA), Amazon Redshift provides browser-based plugins. Use the browser SAML plugin for Okta, PingOne, and the browser Azure AD plugin for Microsoft Azure Active Directory.

With the browser SAML plugin, OAuth authentication flows like this:



1. A user tries to log in.
2. The plugin launches a local server to listen to incoming connections on the localhost.
3. The plugin launches a web browser to request a SAML response over HTTPS from the specified single sign-on login URL federated identity provider endpoint.
4. The web browser follows the link and prompts the user to enter credentials.
5. After the user authenticates and grants consent, the federated identity provider endpoint returns a SAML response over HTTPS to the URI indicated by `redirect_uri`.
6. The web browser moves the response message with the SAML response to the indicated `redirect_uri`.
7. The local server accepts the incoming connection and the plugin retrieves the SAML response and passes it to Amazon Redshift.

With the browser Azure AD plugin, SAML authentication flows like this:



1. A user tries to log in.
2. The plugin launches a local server to listen to incoming connections on the localhost.
3. The plugin launches a web browser to request an authorization code from the Azure AD `oauth2/authorize` endpoint.
4. The web browser follows the generated link over HTTPS and prompts the user to enter credentials. The link is generated using configuration properties, such as tenant and client_id.
5. After the user authenticates and grants consent, the Azure AD `oauth2/authorize` endpoint returns and sends a response over HTTPS with the authorization code to the indicated `redirect_uri`.
6. The web browser moves the response message with the SAML response to the indicated `redirect_uri`.
7. The local server accepts the incoming connection and the plugin requests and retrieves the authorization code and sends a POST request to the Azure AD `oauth2/token` endpoint.
8. The Azure AD `oauth2/token` endpoint returns a response with an access token to the indicated `redirect_uri`.
9. The plugin retrieves the SAML response and passes it to Amazon Redshift.

See the following sections:

- Active Directory Federation Services (AD FS)

  For more information, see Setting up JDBC or ODBC single sign-on authentication.

- PingOne (Ping)

  Ping is supported only with the predetermined PingOne IdP Adapter using Forms authentication.

  For more information, see Setting up JDBC or ODBC single sign-on authentication.

- Okta

  Okta is supported only for the Okta-supplied application used with the AWS Management Console.

  For more information, see Setting up JDBC or ODBC single sign-on authentication.

- Microsoft Azure Active Directory

  For more information, see Setting up JDBC or ODBC single sign-on authentication.

Plugin options
Plugin options

To use a SAML-based credentials provider plugin, specify the following options using JDBC or ODBC options or in a named profile. If `plugin_name` isn't specified, the other options are ignored.

Option	Description

`plugin_name`	For JDBC, the class name that implements a credentials provider. Specify one of the following:
	• For Active Directory Federation Services
	`com.amazon.redshift.plugin.AdfsCredentialsProvider`
	• For Okta
	`com.amazon.redshift.plugin.OktaCredentialsProvider`
	• For PingFederate

Option	Description

	`com.amazon.redshift.plugin.PingCredentialsProvider`
	• For Microsoft Azure Active Directory
	`com.amazon.redshift.plugin.AzureCredentialsProvider`
	• For SAML MFA
	`com.amazon.redshift.plugin.BrowserSamlCredentialsProvider`
	• For Microsoft Azure Active Directory single sign-on with MFA
	`com.amazon.redshift.plugin.BrowserAzureCredentialsProvider`
	
	For ODBC, specify one of the following:
	• For Active Directory Federation Services: `adfs`
	• For Okta: `okta`
	• For PingFederate: `ping`
	• For Microsoft Azure Active Directory: `azure`
	• For SAML MFA: `browser saml`
	• For Microsoft Azure Active Directory single sign-on with MFA: `browser azure ad`

`idp_host`	The name of the corporate identity provider host. This name should not include any slashes ('/'). For an Okta identity provider, the value for `idp_host` should end with `.okta.com .`
`idp_port`	The port used by the identity provider. The default is 443. This port is ignored for Okta.

Option	Description
preferred_role	A role Amazon Resource Name (ARN) from the AttributeValue elements for the Role attribute in the SAML assertion. To find the appropriate value for the preferred role, work with your IdP administrator. For more information, see Step 2: Configure SAML assertions for your IdP.
user	A corporate user name, including the domain when applicable. For example, for Active Directory, the domain name is required in the format *domain**username*.
password	The corporate user's password. We recommend not using this option. Instead, use your SQL client to supply the password.
app_id	An ID for an Okta application. Used only with Okta. The value for app_id follows amazon_aws in the Okta application embed link. To get this value, work with your IdP administrator. The following is an example of an application embed link: https://example.okta.com/home/amazon_aws/0oa2hylwrpM8U Gehd1t7/272
idp_tenant	A tenant used for Azure AD. Used only with Azure.
client_id	A client ID for the Amazon Redshift enterprise application in Azure AD. Used only with Azure.

Generating database credentials for an IAM identity using the Amazon Redshift CLI or API

To programmatically generate temporary database user credentials, Amazon Redshift provides the get-cluster-credentials command for the AWS Command Line Interface (AWS CLI) and the GetClusterCredentials API operation. Or you can configure your SQL client with Amazon Redshift JDBC or ODBC drivers that manage the process of calling the GetClusterCredentials operation, retrieving the database user credentials, and establishing a connection between your SQL client and your Amazon Redshift database. For more information, see JDBC and ODBC options for creating database user credentials.

> **Note**
>
> We recommend using the Amazon Redshift JDBC or ODBC drivers to generate database user credentials.

In this section, you can find steps to programmatically call the `GetClusterCredentials` operation or get-cluster-credentials command, retrieve database user credentials, and connect to the database.

To generate and use temporary database credentials

1. Create or modify a user or role with the required permissions. For more information about IAM permissions, see Step 3: Create an IAM role with permissions to call GetClusterCredentialsWithIAM or GetClusterCredentials.
2. As a user or role you authorized in the previous step, run the get-cluster-credentials CLI command or call the `GetClusterCredentials` API operation and provide the following values:
   - **Cluster identifier** – The name of the cluster that contains the database.
   - **Database user name** – The name of an existing or new database user.
     - If the user doesn't exist in the database and AutoCreate is true, a new user is created with PASSWORD disabled.
     - If the user doesn't exist, and AutoCreate is false, the request fails.
     - For this example, the database user name is `temp_creds_user`.
   - **Autocreate** – (Optional) Create a new user if the database user name doesn't exist.
   - **Database name** – (Optional) The name of the database that the user is authorized to log on to. If database name isn't specified, the user can log on to any cluster database.
   - **Database groups** – (Optional) A list of existing database user groups. Upon successful login, the database user is added to the specified user groups. If no group is specified, the user has only PUBLIC permissions. The user group names must match the dbgroup resources ARNs specified in the IAM policy attached to the user or role.
   - **Expiration time** – (Optional) The time, in seconds, until the temporary credentials expire. You can specify a value between 900 seconds (15 minutes) and 3600 seconds (60 minutes). The default is 900 seconds.

3. Amazon Redshift verifies that the user has permission to call the `GetClusterCredentials` operation with the specified resources.
4. Amazon Redshift returns a temporary password and the database user name.

   The following example uses the Amazon Redshift CLI to generate temporary database credentials for an existing user named `temp_creds_user`.

   ```
   aws redshift get-cluster-credentials --cluster-identifier examplecluster --db-user
    temp_creds_user --db-name exampledb --duration-seconds 3600
   ```

   The result is as follows.

   ```
   {
     "DbUser": "IAM:temp_creds_user",
     "Expiration": "2016-12-08T21:12:53Z",
     "DbPassword": "EXAMPLEjArE3hcnQj8zt4XQj9Xtma8oxYEM8OyxpDHwXVPyJYBDm/
   gqX2Eeaq6P3DgTzgPg=="
   }
   ```

   The following example uses the Amazon Redshift CLI with autocreate to generate temporary database credentials for a new user and add the user to the group `example_group`.

   ```
   aws redshift get-cluster-credentials --cluster-identifier examplecluster --db-user
    temp_creds_user --auto-create --db-name exampledb --db-groups example_group --
   duration-seconds 3600
   ```

   The result is as follows.

   ```
   {
     "DbUser": "IAMA:temp_creds_user:example_group",
     "Expiration": "2016-12-08T21:12:53Z",
     "DbPassword": "EXAMPLEjArE3hcnQj8zt4XQj9Xtma8oxYEM8OyxpDHwXVPyJYBDm/
   gqX2Eeaq6P3DgTzgPg=="
   }
   ```

5. Establish a Secure Socket Layer (SSL) authentication connection with the Amazon Redshift cluster and send a login request with the user name and password from the `GetClusterCredentials` response. Include the `IAM:` or `IAMA:` prefix with the user name, for example `IAM:temp_creds_user` or `IAMA:temp_creds_user`.

> **Important**
>
> Configure your SQL client to require SSL. Otherwise, if your SQL client automatically tries to connect with SSL, it can fall back to non-SSL if there is any kind of failure. In that case, the first connection attempt might fail because the credentials are expired or invalid, then a second connection attempt fails because the connection is not SSL. If that occurs, the first error message might be missed. For more information about connecting to your cluster using SSL, see Configuring security options for connections.

6. If the connection doesn't use SSL, the connection attempt fails.
7. The cluster sends an `authentication` request to the SQL client.
8. The SQL client then sends the temporary password to the cluster.
9. If the password is valid and has not expired, the cluster completes the connection.

Setting up JDBC or ODBC single sign-on authentication

You can leverage external identity providers (IdPs) to authenticate and authorize users accessing your Amazon Redshift cluster, simplifying user management and enhancing security. This enables centralized user management, role-based access control, and auditing capabilities across multiple services. Common use cases include streamlining authentication for diverse user groups, enforcing consistent access policies, and meeting regulatory requirements.

The following pages guide you through configuring IdP integration with your Redshift cluster. For more information about configuring AWS as a service provider for the IdP, see Configuring Your SAML 2.0 IdP with Relying Party Trust and Adding Claims in the *IAM User Guide*.

AD FS

This tutorial shows you how you can use AD FS as an identity provider (IdP) to access your Amazon Redshift cluster.

Step 1: Set up AD FS and your AWS account to trust each other

The following procedure describes how to set up a trust relationship.

1. Create or use an existing Amazon Redshift cluster for your AD FS users to connect to. To configure the connection, certain properties of this cluster are needed, such as the cluster identifier. For more information, see Creating a Cluster.

2. Set up AD FS to control Amazon Redshift access on the Microsoft Management Console:

    1. Choose **ADFS 2.0**, and then choose **Add Relying Party Trust**. On the **Add Relying Party Trust Wizard** page, choose **Start**.
    2. On the **Select Data Source** page, choose **Import data about the relying party published online or on a local network**.
    3. For **Federation metadata address (host name or URL)**, enter **`https://signin.aws.amazon.com/saml-metadata.xml`**. The metadata XML file is a standard SAML metadata document that describes AWS as a relying party.
    4. On the **Specify Display Name** page, enter a value for **Display name**.
    5. On the **Choose Issuance Authorization Rules** page, choose an issuance authorization rule to either permit or deny all users to access this relying party.
    6. On the **Ready to Add Trust** page, review your settings.
    7. On the **Finish** page, choose **Open the Edit Claim Rules dialog for this relying party trust when the wizard closes**.
    8. On the context (right-click) menu, choose **Relying Party Trusts**.
    9. For your relying party, open the context (right-click) menu and choose **Edit Claim Rules**. On the **Edit Claim Rules** page, choose **Add Rule**.
    10. For **Claim rule template**, choose **Transform an Incoming Claim**, and then on the **Edit Rule – NameId** page, do the following:
        - For **Claim rule name**, enter **NameId**.
        - For **Incoming claim name**, choose **Windows Account Name**.
        - For **Outgoing claim name**, choose **Name ID**.
        - For **Outgoing name ID format**, choose **Persistent Identifier**.
        - Choose **Pass through all claim values**.
    11. On the **Edit Claim Rules** page, choose **Add Rule**. On the **Select Rule Template** page, for **Claim rule template**, choose **Send LDAP Attributes as Claims**.
    12. On the **Configure Rule** page, do the following:
        - For **Claim rule name**, enter **RoleSessionName**.
        - For **Attribute store**, choose **Active Directory**.
        - For **LDAP Attribute**, choose **Email Addresses**.
        - For **Outgoing Claim Type**, choose **https://aws.amazon.com/SAML/Attributes/RoleSessionName**.

13. On the **Edit Claim Rules** page, choose **Add Rule**. On the **Select Rule Template** page, for **Claim rule template**, choose **Send Claims Using a Custom Rule**.
14. On the **Edit Rule – Get AD Groups** page, for **Claim rule name**, enter **Get AD Groups**.
15. For **Custom rule**, enter the following.

```
c:[Type ==
                                    "http://schemas.microsoft.com/ws/2008/06/
identity/claims/windowsaccountname",
                                    Issuer == "AD AUTHORITY"] => add(store =
 "Active Directory",
                                    types = ("http://temp/variable"), query =
 ";tokenGroups;{0}",
                                    param = c.Value);
```

16. On the **Edit Claim Rules** page, choose **Add Rule**. On the **Select Rule Template** page, for **Claim rule template**, choose **Send Claims Using a Custom Rule**.
17. On the **Edit Rule – Roles** page, for **Claim rule name**, type **Roles**.
18. For **Custom rule,** enter the following.

```
c:[Type == "http://temp/variable", Value =~ "(?i)^AWS-"] =>
 issue(Type = "https://aws.amazon.com/SAML/Attributes/Role", Value =
 RegExReplace(c.Value, "AWS-", "arn:aws:iam::123456789012:saml-provider/
ADFS,arn:aws:iam::123456789012:role/ADFS-"));
```

Note the ARNs of the SAML provider and role to assume. In this example, `arn:aws:iam:123456789012:saml-provider/ADFS` is the ARN of the SAML provider and `arn:aws:iam:123456789012:role/ADFS-` is the ARN of the role.

3. Make sure that you have downloaded the `federationmetadata.xml` file. Check that the document contents do not have invalid characters. This is the metadata file you use when configuring the trust relationship with AWS.
4. Create an IAM SAML identity provider on the IAM console. The metadata document. that you provide is the federation metadata XML file that you saved when you set up Azure Enterprise Application. For detailed steps, see Creating and Managing an IAM Identity Provider (Console) in the *IAM User Guide*.
5. Create an IAM role for SAML 2.0 federation on the IAM console. For detailed steps, see Creating a Role for SAML in the *IAM User Guide*.

6. Create an IAM policy that you can attach to the IAM role that you created for SAML 2.0 federation on the IAM console. For detailed steps, see Creating IAM Policies (Console) in the *IAM User Guide*. For an Azure AD example, see Setting up JDBC or ODBC single sign-on authentication.

Step 2: Set up JDBC or ODBC for authentication to AD FS

JDBC

The following procedure describes how to set up a JDBC relationship to AD FS.

- Configure your database client to connect to your cluster through JDBC using AD FS single sign-on.

  You can use any client that uses a JDBC driver to connect using AD FS single sign-on or use a language like Java to connect using a script. For installation and configuration information, see Configuring a connection for JDBC driver version 2.x for Amazon Redshift.

  For example, you can use SQLWorkbench/J as the client. When you configure SQLWorkbench/J, the URL of your database uses the following format.

  ```
  jdbc:redshift:iam://cluster-identifier:us-west-1/dev
  ```

  If you use SQLWorkbench/J as the client, take the following steps:

  a. Start SQL Workbench/J. In the **Select Connection Profile** page, add a **Profile Group**, for example **ADFS**.

  b. For **Connection Profile**, enter your connection profile name, for example **ADFS**.

  c. Choose **Manage Drivers**, and choose **Amazon Redshift**. Choose the **Open Folder** icon next to **Library**, then choose the appropriate JDBC .jar file.

  d. On the **Select Connection Profile** page, add information to the connection profile as follows:

  - For **User**, enter your AD FS user name. This is the user name of the account that you are using for single sign-on that has permission to the cluster that you are trying to authenticate using.
  - For **Password**, enter your AD FS password.

- For **Drivers**, choose **Amazon Redshift (com.amazon.redshift.jdbc.Driver)**.
- For **URL**, enter **`jdbc:redshift:iam://`*`your-cluster-identifier`*`:`*`your-cluster-region`*`/`*`your-database-name`***.

e. Choose **Extended Properties**. For **plugin_name**, enter **`com.amazon.redshift.plugin.AdfsCredentialsProvider`**. This value specifies to the driver to use AD FS single sign-on as the authentication method.

ODBC

To set up ODBC for authentication to AD FS

- Configure your database client to connect to your cluster through ODBC using AD FS single sign-on.

  Amazon Redshift provides ODBC drivers for Linux, Windows, and macOS operating systems. Before you install an ODBC driver, determine whether your SQL client tool is 32-bit or 64-bit. Install the ODBC driver that matches the requirements of your SQL client tool.

  On Windows, in the **Amazon Redshift ODBC Driver DSN Setup** page, under **Connection Settings**, enter the following information:

  - For **Data Source Name**, enter ***`your-DSN`***. This specifies the data source name used as the ODBC profile name.
  - For **Auth type**, choose **Identity Provider: SAML**. This is the authentication method that the ODBC driver uses to authenticate using AD FS single sign-on.
  - For **Cluster ID**, enter ***`your-cluster-identifier`***.
  - For **Region**, enter ***`your-cluster-region`***.
  - For **Database**, enter ***`your-database-name`***.
  - For **User**, enter ***`your-adfs-username`***. This is the user name for the AD FS account that you are using for single sign-on that has permission to the cluster that you're trying to authenticate using. Use this only for **Auth type** is **Identity Provider: SAML**.
  - For **Password**, enter ***`your-adfs-password`***. Use this only for **Auth type** is **Identity Provider: SAML**.

  On macOS and Linux, edit the `odbc.ini` file as follows:

> **Note**
>
> All entries are case-insensitive.

- For **clusterid**, enter ***your-cluster-identifier***. This is the name of the created Amazon Redshift cluster.
- For **region**, enter ***your-cluster-region***. This is the AWS Region of the created Amazon Redshift cluster.
- For **database**, enter ***your-database-name***. This is the name of the database that you're trying to access on the Amazon Redshift cluster.
- For **locale**, enter **en-us**. This is the language that error messages display in.
- For **iam**, enter **1**. This value specifies to the driver to authenticate using IAM credentials.
- For **plugin_name**, do one of the following:
  - For AD FS single sign-on with MFA configuration, enter **BrowserSAML**. This is the authentication method that the ODBC driver uses to authenticate to AD FS.
  - For AD FS single sign-on configuration, enter **ADFS**. This is the authentication method that the ODBC driver uses to authenticate using Azure AD single sign-on.
- For **uid**, enter ***your-adfs-username***. This is the user name of the Microsoft Azure account that you are using for single sign-on that has permission to the cluster you are trying to authenticate against. Use this only for **plugin_name** is **ADFS**.
- For **pwd**, enter ***your-adfs-password***. Use this only for **plugin_name** is **ADFS**.

On macOS and Linux, also edit the profile settings to add the following exports.

```
export ODBCINI=/opt/amazon/redshift/Setup/odbc.ini
```

```
export ODBCINSTINI=/opt/amazon/redshift/Setup/odbcinst.ini
```

Azure

You can use Microsoft Azure AD as an identity provider (IdP) to access your Amazon Redshift cluster. This tutorial shows you how you can use Azure as an identity provider (IdP) to access your Amazon Redshift cluster.

Watch the following video to learn how to federate Amazon Redshift access with Microsoft Azure AD single sign-on: Federating Amazon Redshift access with Microsoft Azure AD single sign-on.

Step 1: Set up Azure and your AWS account to trust each other

The following procedure describes how to set up a trust relationship.

To set up Azure AD and your AWS account to trust each other

1. Create or use an existing Amazon Redshift cluster for your Azure AD users to connect to. To configure the connection, certain properties of this cluster are needed, such as the cluster identifier. For more information, see Creating a Cluster.
2. Set up an Azure Active Directory, groups, users used for AWS on the Microsoft Azure portal.
3. Add Amazon Redshift as an enterprise application on the Microsoft Azure portal to use for single sign-on to the AWS Console and federated login to Amazon Redshift. Choose **Enterprise application**.
4. Choose **+New application**. The Add an application page appears.
5. Search for **AWS** in the search field.
6. Choose **Amazon Web Services (AWS)** and choose **Add**. This creates the AWS application.
7. Under **Manage**, choose **Single sign-on**.
8. Choose **SAML**. The Amazon Web Services (AWS) | SAML-based Sign-on page appears.
9. Choose **Yes** to proceed to the Set up Single Sign-On with SAML page. This page shows the list of pre-configured single sign-on related attributes.
10. For **Basic SAML Configuration**, choose the edit icon and choose **Save**.
11. When you are configuring for more than one application, provide an identifier value. For example, enter ***https://signin.aws.amazon.com/saml#2***. Note that from the second application onwards, use this format with a # sign to specify a unique SPN value.
12. In the **User Attributes and Claims** section, choose the edit icon.

    By default, the Unique User Identifier (UID), Role, RoleSessionName, and SessionDuration claims are pre-configured.

13. Choose **+ Add new claim** to add a claim for database users.

    For **Name**, enter **`DbUser`**.

    For **Namespace**, enter **`https://redshift.amazon.com/SAML/Attributes`**.

    For **Source**, choose **Attribute**.

    For **Source attribute**, choose **user.userprincipalname**. Then, choose **Save**.

14. Choose **+ Add new claim** to add a claim for AutoCreate.

    For **Name**, enter **`AutoCreate`**.

    For **Namespace**, enter **`https://redshift.amazon.com/SAML/Attributes`**.

    For **Source**, choose **Attribute**.

    For **Source attribute**, choose **"true"**. Then, choose **Save**.

    Here, *123456789012* is your AWS account, *AzureSSO* is an IAM role you created, and *AzureADProvider* is the IAM provider.

Claim name	Value
Unique user identifier (name ID)	user.userprincipalname
https://aws.amazon.com/SAML/Attributes/SessionDuration	"900"
https://aws.amazon.com/SAML/Attributes/Role	arn:aws:iam::*123456789012* :role/*AzureSSO*,arn:aws:iam::*123456789012* :saml-provider/*AzureADProvider*
https://aws.amazon.com/SAML/Attributes/RoleSessionName	user.userprincipalname
https://redshift.amazon.com/SAML/Attributes/AutoCreate	"true"

Claim name	Value
https://redshift.amazon.com/SAML/Attributes/DbGroups	user.assignedroles
https://redshift.amazon.com/SAML/Attributes/DbUser	user.userprincipalname

15. Under **App Registration > *your-application-name* > Authentication**, add **Mobile And Desktop Application**. Specify the URL as http://localhost/redshift/.
16. In the **SAML Signing Certificate** section, choose **Download** to download and save the federation metadata XML file for use when you create an IAM SAML identity provider. This file is used to create the single sign-on federated identity.
17. Create an IAM SAML identity provider on the IAM console. The metadata document that you provide is the federation metadata XML file that you saved when you set up Azure Enterprise Application. For detailed steps, see Creating and Managing an IAM Identity Provider (Console) in the *IAM User Guide*.
18. Create an IAM role for SAML 2.0 federation on the IAM console. For detailed steps, see Creating a Role for SAML in the *IAM User Guide*.
19. Create an IAM policy that you can attach to the IAM role that you created for SAML 2.0 federation on the IAM console. For detailed steps, see Creating IAM Policies (Console) in the *IAM User Guide*.

    Modify the following policy (in JSON format) for your environment:

    - Substitute the AWS Region of your cluster for *us-west-1*.
    - Substitute your AWS account for *123456789012*.
    - Substitute your cluster identifier (or * for all clusters) for *cluster-identifier*.
    - Substitute your database (or * for all databases) for *dev*.
    - Substitute the unique identifier of your IAM role for *AROAJ2UCCR6DPCEXAMPLE*.
    - Substitute your tenant or company email domain for *example.com*.
    - Substitute the database group that you plan to assign the user to for *my_dbgroup*.

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "redshift:GetClusterCredentials",
            "Resource": [
                "arn:aws:redshift:us-west-1:123456789012:dbname:cluster-
identifier/dev",
                "arn:aws:redshift:us-west-1:123456789012:dbuser:cluster-identifier/
${redshift:DbUser}",
                "arn:aws:redshift:us-west-1:123456789012:cluster:cluster-
identifier"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:userid": "AROAJ2UCCR6DPCEXAMPLE:
${redshift:DbUser}@example.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "redshift:CreateClusterUser",
            "Resource": "arn:aws:redshift:us-west-1:123456789012:dbuser:cluster-
identifier/${redshift:DbUser}"
        },
        {
            "Effect": "Allow",
            "Action": "redshift:JoinGroup",
            "Resource": "arn:aws:redshift:us-west-1:123456789012:dbgroup:cluster-
identifier/my_dbgroup"
        },
        {
            "Effect": "Allow",
            "Action": [
                "redshift:DescribeClusters",
                "iam:ListRoles"
            ],
            "Resource": "*"
        }
    ]
}
```

This policy grants permissions as follows:

- The first section grants permission to the `GetClusterCredentials` API operation to get temporary credentials for the specified cluster. In this example, the resource is *`cluster-identifier`* with database *`dev`*, in account *`123456789012`*, and in AWS Region *`us-west-1`*. The `${redshift:DbUser}` clause allows only users that match the `DbUser` value specified in Azure AD to connect.
- The condition clause enforces that only certain users get temporary credentials. These are users under the role specified by the role unique ID *`AROAJ2UCCR6DPCEXAMPLE`* in the IAM account identified by an email address in your company's email domain. For more information about unique IDs, see Unique IDs in the *IAM User Guide*.

  Your setup with your IdP (in this case, Azure AD) determines how the condition clause is written. If your employee's email is `johndoe@example.com`, first set `${redshift:DbUser}` to the super field that matches the employee's user name `johndoe`. Then, to make this condition work, set the AWS SAML `RoleSessionName` field to the super field that matches the employee's email `johndoe@example.com`. When you take this approach, consider the following:

  - If you set `${redshift:DbUser}` to be the employee's email, then remove the `@example.com` in the example JSON to match the `RoleSessionName`.
  - If you set the `RoleSessionId` to be just the employee's user name, then remove the `@example.com` in the example to match the `RoleSessionName`.
  - In the example JSON, the `${redshift:DbUser}` and `RoleSessionName` are both set to the employee's email. This example JSON uses the Amazon Redshift database user name with `@example.com` to sign the user in to access the cluster.
- The second section grants permission to create a `dbuser` name in the specified cluster. In this example JSON, it restricts creation to `${redshift:DbUser}`.
- The third section grants permission to specify which `dbgroup` a user can join. In this example JSON, a user can join the `my_dbgroup` group in the specified cluster.
- The fourth section grants permission to actions the user can do on all resources. In this example JSON, it allows users to call `redshift:DescribeClusters` to get cluster information such as the cluster endpoint, AWS Region, and port. It also allows users to call `iam:ListRoles` to check which roles a user can assume.

Step 2: Set up JDBC or ODBC for authentication to Azure

JDBC

To set up JDBC for authentication to Microsoft Azure AD

- Configure your database client to connect to your cluster through JDBC using your Azure AD single sign-on.

  You can use any client that uses a JDBC driver to connect using Azure AD single sign-on or use a language like Java to connect using a script. For installation and configuration information, see Configuring a connection for JDBC driver version 2.x for Amazon Redshift.

  For example, you can use SQLWorkbench/J as the client. When you configure SQLWorkbench/J, the URL of your database uses the following format.

  ```
  jdbc:redshift:iam://cluster-identifier:us-west-1/dev
  ```

  If you use SQLWorkbench/J as the client, take the following steps:

  a. Start SQL Workbench/J. On the **Select Connection Profile** page, add a **Profile Group** called **`AzureAuth`**.

  b. For **Connection Profile**, enter **`Azure`**.

  c. Choose **Manage Drivers**, and choose **Amazon Redshift**. Choose the **Open Folder** icon next to **Library**, then choose the appropriate JDBC .jar file.

  d. On the **Select Connection Profile** page, add information to the connection profile as follows:

  - For **User**, enter your Microsoft Azure user name. This is the user name of the Microsoft Azure account that you are using for single sign-on that has permission to the cluster that you are trying to authenticate using.
  - For **Password**, enter your Microsoft Azure password.
  - For **Drivers**, choose **Amazon Redshift (com.amazon.redshift.jdbc.Driver)**.
  - For **URL**, enter **`jdbc:redshift:iam://`*`your-cluster-identifier`*`:`*`your-cluster-region`*`/`*`your-database-name`*.

  e. Choose **Extended Properties** to add additional information to the connection properties, as described following.

For Azure AD single sign-on configuration, add additional information as follows:

- For **plugin_name**, enter **`com.amazon.redshift.plugin.AzureCredentialsProvider`**. This value specifies to the driver to use Azure AD Single Sign-On as the authentication method.
- For **idp_tenant**, enter ***`your-idp-tenant`***. Used only for Microsoft Azure AD. This is the tenant name of your company configured on Azure AD. This value can either be the tenant name or the tenant unique ID with hyphens.
- For **client_secret**, enter ***`your-azure-redshift-application-client-secret`***. Used only for Microsoft Azure AD. This is your client secret of the Amazon Redshift application that you created when setting up your Azure Single Sign-On configuration. This is only applicable to the com.amazon.redshift.plugin.AzureCredentialsProvider plugin.
- For **client_id**, enter ***`your-azure-redshift-application-client-id`***. Used only for Microsoft Azure AD. This is the client ID (with hyphens) of the Amazon Redshift application that you created when setting up your Azure Single Sign-On configuration.

For Azure AD single sign-on with MFA configuration, add additional information to the connection properties as follows:

- For **plugin_name**, enter **`com.amazon.redshift.plugin.BrowserAzureCredentialsProvider`**. This value specifies to the driver to use Azure AD single sign-on with MFA as the authentication method.
- For **idp_tenant**, enter ***`your-idp-tenant`***. Used only for Microsoft Azure AD. This is the tenant name of your company configured on Azure AD. This value can either be the tenant name or the tenant unique ID with hyphens.
- For **client_id**, enter ***`your-azure-redshift-application-client-id`***. This option is used only for Microsoft Azure AD. This is the client ID (with hyphens) of the Amazon Redshift application that you created when setting up your Azure AD single sign-on with MFA configuration.
- For **listen_port**, enter ***`your-listen-port`***. This is the port that local server is listening to. The default is 7890.

- For **idp_response_timeout**, enter ***the-number-of-seconds***. This is the number of seconds to wait before timing out when the IdP server sends back a response. The minimum number of seconds must be 10. If establishing the connection takes longer than this threshold, then the connection is aborted.

ODBC

To set up ODBC for authentication to Microsoft Azure AD

- Configure your database client to connect to your cluster through ODBC using your Azure AD single sign-on.

  Amazon Redshift provides ODBC drivers for Linux, Windows, and macOS operating systems. Before you install an ODBC driver, determine whether your SQL client tool is 32-bit or 64-bit. Install the ODBC driver that matches the requirements of your SQL client tool.

  On Windows, in the **Amazon Redshift ODBC Driver DSN Setup** page, under **Connection Settings**, enter the following information:

  - For **Data Source Name**, enter ***your-DSN***. This specifies the data source name used as the ODBC profile name.
  - For **Auth type** for Azure AD single sign-on configuration, choose **`Identity Provider: Azure AD`**. This is the authentication method that the ODBC driver uses to authenticate using Azure single sign-on.
  - For **Auth type** for Azure AD single sign-on with MFA configuration, choose **`Identity Provider: Browser Azure AD`**. This is the authentication method that the ODBC driver uses to authenticate using Azure single sign-on with MFA.
  - For **Cluster ID**, enter ***your-cluster-identifier***.
  - For **Region**, enter ***your-cluster-region***.
  - For **Database**, enter ***your-database-name***.
  - For **User**, enter ***your-azure-username***. This is the user name for the Microsoft Azure account that you are using for single sign-on that has permission to the cluster that you're trying to authenticate using. Use this only for **Auth Type** is **Identity Provider: Azure AD**.
  - For **Password**, enter ***your-azure-password***. Use this only for **Auth Type** is **Identity Provider: Azure AD**.

- For **IdP Tenant**, enter ***your-idp-tenant***. This is the tenant name of your company configured on your IdP (Azure). This value can either be the tenant name or the tenant unique ID with hyphens.
- For **Azure Client Secret**, enter ***your-azure-redshift-application-client-secret***. This is the client secret of the Amazon Redshift application that you created when setting up your Azure single sign-on configuration.
- For **Azure Client ID**, enter ***your-azure-redshift-application-client-id***. This is the client ID (with hyphens) of the Amazon Redshift application that you created when setting up your Azure single sign-on configuration.
- For **Listen Port**, enter ***your-listen-port***. This is the default listen port that local server is listening to. The default is 7890. This applies only to the Browser Azure AD plugin.
- For **Response Timeout**, enter ***the-number-of-seconds***. This is the number of seconds to wait before timing out when the IdP server sends back a response. The minimum number of seconds must be 10. If establishing the connection takes longer than this threshold, then the connection is aborted. This option applies only to the Browser Azure AD plugin.

On macOS and Linux, edit the `odbc.ini` file as follows:

> **Note**
>
> All entries are case-insensitive.

- For **clusterid**, enter ***your-cluster-identifier***. This is the name of the created Amazon Redshift cluster.
- For **region**, enter ***your-cluster-region***. This is the AWS Region of the created Amazon Redshift cluster.
- For **database**, enter ***your-database-name***. This is the name of the database that you're trying to access on the Amazon Redshift cluster.
- For **locale**, enter **en-us**. This is the language that error messages display in.
- For **iam**, enter **1**. This value specifies to the driver to authenticate using IAM credentials.

- For **plugin_name** for Azure AD single sign-on configuration, enter **`AzureAD`**. This specifies to the driver to use Azure Single Sign-On as the authentication method.
- For **plugin_name** for Azure AD single sign-on with MFA configuration, enter **`BrowserAzureAD`**. This specifies to the driver to use Azure Single Sign-On with MFA as the authentication method.
- For **uid**, enter ***`your-azure-username`***. This is the user name of the Microsoft Azure account you are using for single sign-on that has permission to the cluster you are trying to authenticate against. Use this only for **plugin_name** is **AzureAD**.
- For **pwd**, enter ***`your-azure-password`***. Use this only for **plugin_name** is **AzureAD**.
- For **idp_tenant**, enter ***`your-idp-tenant`***. This is the tenant name of your company configured on your IdP (Azure). This value can either be the tenant name or the tenant unique ID with hyphens.
- For **client_secret**, enter ***`your-azure-redshift-application-client-secret`***. This is the client secret of the Amazon Redshift application that you created when setting up your Azure single sign-on configuration.
- For **client_id**, enter ***`your-azure-redshift-application-client-id`***. This is the client ID (with hyphens) of the Amazon Redshift application that you created when setting up your Azure single sign-on configuration.
- For **listen_port**, enter ***`your-listen-port`***. This is the port that local server is listening to. The default is 7890. This applies to the Browser Azure AD plugin.
- For **idp_response_timeout**, enter ***`the-number-of-seconds`***. This is the specified period of time in seconds to wait for response from Azure. This option applies to the Browser Azure AD plugin.

On macOS and Linux, also edit the profile settings to add the following exports.

```
export ODBCINI=/opt/amazon/redshift/Setup/odbc.ini
```

```
export ODBCINSTINI=/opt/amazon/redshift/Setup/odbcinst.ini
```

Troubleshooting

To troubleshoot issues with the Browser Azure AD plugin, consider the following.

- To use the Browser Azure AD plugin, you must set the reply URL specified in the request to match the reply URL configured for your application. Navigate to the **Set up Single Sign-On with SAML** page on the Microsoft Azure portal. Then check the **Reply URL** is set to http://localhost/redshift/.
- If you get an IdP tenant error, verify that the **IdP Tenant** name matches the domain name you initially used to set up the Active Directory in Microsoft Azure.

  On Windows, navigate to the **Connection Settings** section of the **Amazon Redshift ODBC DSN Setup** page. Then check the tenant name of your company configured on your IdP (Azure) matches the domain name you initially used to set up the Active Directory in Microsoft Azure.

  On macOS and Linux, find the *odbc.ini* file. Then check the tenant name of your company configured on your IdP (Azure) matches the domain name you initially used to set up the Active Directory in Microsoft Azure.
- If you get an error that the reply URL specified in the request does not match the reply URLs configured for your application, verify that the **Redirect URIs** is the same as the reply URL.

  Navigate to the **App registration** page of your application on the Microsoft Azure portal. Then check the Redirect URIs matches the reply URL.
- If you get the unexpected response: unauthorized error, verify that you completed the **Mobile and desktop applications** configuration.

  Navigate to the **App registration** page of your application on the Microsoft Azure portal. Then navigate to **Authentication** and check that you configured **Mobile and desktop applications** to use http://localhost/redshift/ as the redirect URIs.

Ping Identity

You can use Ping Identity as an identity provider (IdP) to access your Amazon Redshift cluster. This tutorial shows you how you can use Ping Identity as an identity provider (IdP) to access your Amazon Redshift cluster.

Step 1: Set up Ping Identity and your AWS account to trust each other

The following procedure describes how to set up a trust relationship using the PingOne portal.

To set up Ping Identity and your AWS account to trust each other

1. Create or use an existing Amazon Redshift cluster for your Ping Identity users to connect to. To configure the connection, certain properties of this cluster are needed, such as the cluster identifier. For more information, see Creating a Cluster.
2. Add Amazon Redshift as a new SAML application on the PingOne portal. For detailed steps, see the Ping Identity documentation.
   1. Go to **My Applications**.
   2. Under **Add Application**, choose **New SAML Application**.
   3. For **Application Name**, enter **`Amazon Redshift`**.
   4. For **Protocol Version**, choose **SAML v2.0**.
   5. For **Category**, choose ***`your-application-category`***.
   6. For **Assertion Consumer Service (ACS)**, type ***`your-redshift-local-host-url`***. This is the local host and port that the SAML assertion redirects to.
   7. For **Entity ID**, enter `urn:amazon:webservices`.
   8. For **Signing**, choose **Sign Assertion**.
   9. In the **SSO Attribute Mapping** section, create the claims as shown in the following table.

Application attribute	Identity bridge attribute of literal value
https://aws.amazon.com/SAML/Attributes/Role	arn:aws:iam::*123456789012* :role/*Ping*,arn:aws:iam::*123456789012* :saml-provider/*PingProvider*
https://aws.amazon.com/SAML/Attributes/RoleSessionName	email
https://redshift.amazon.com/SAML/Attributes/AutoCreate	"true"
https://redshift.amazon.com/SAML/Attributes/DbUser	email
https://redshift.amazon.com/SAML/Attributes/DbGroups	The groups in the “DbGroups” attributes contain the @directory prefix. To remove

Application attribute	Identity bridge attribute of literal value
	this, in **Identity bridge**, enter **memberOf**. In **Function**, choose **ExtractByRegularExpression**. In **Expression**, enter **(.*)[\@](?:.*)**.

3. For **Group Access**, set up the following group access, if needed:
   - **https://aws.amazon.com/SAML/Attributes/Role**
   - **https://aws.amazon.com/SAML/Attributes/RoleSessionName**
   - **https://redshift.amazon.com/SAML/Attributes/AutoCreate**
   - **https://redshift.amazon.com/SAML/Attributes/DbUser**
4. Review your setup and make changes, if necessary.
5. Use the **Initiate Single Sign-On (SSO) URL** as the login URL for the Browser SAML plugin.
6. Create an IAM SAML identity provider on the IAM console. The metadata document that you provide is the federation metadata XML file that you saved when you set up Ping Identity. For detailed steps, see Creating and Managing an IAM Identity Provider (Console) in the *IAM User Guide*.
7. Create an IAM role for SAML 2.0 federation on the IAM console. For detailed steps, see Creating a Role for SAML in the *IAM User Guide*.
8. Create an IAM policy that you can attach to the IAM role that you created for SAML 2.0 federation on the IAM console. For detailed steps, see Creating IAM Policies (Console) in the *IAM User Guide*. For an Azure AD example, see Setting up JDBC or ODBC single sign-on authentication.

Step 2: Set up JDBC or ODBC for authentication to Ping Identity

JDBC

To set up JDBC for authentication to Ping Identity

- Configure your database client to connect to your cluster through JDBC using Ping Identity single sign-on.

You can use any client that uses a JDBC driver to connect using Ping Identity single sign-on or use a language like Java to connect using a script. For installation and configuration information, see Configuring a connection for JDBC driver version 2.x for Amazon Redshift.

For example, you can use SQLWorkbench/J as the client. When you configure SQLWorkbench/J, the URL of your database uses the following format.

```
jdbc:redshift:iam://cluster-identifier:us-west-1/dev
```

If you use SQLWorkbench/J as the client, take the following steps:

a. Start SQL Workbench/J. In the **Select Connection Profile** page, add a **Profile Group**, for example **`Ping`**.

b. For **Connection Profile**, enter ***`your-connection-profile-name`***, for example **`Ping`**.

c. Choose **Manage Drivers**, and choose **Amazon Redshift**. Choose the **Open Folder** icon next to **Library**, then choose the appropriate JDBC .jar file.

d. On the **Select Connection Profile** page, add information to the connection profile as follows:

- For **User**, enter your PingOne user name. This is the user name of the PingOne account that you are using for single sign-on that has permission to the cluster that you are trying to authenticate using.
- For **Password**, enter your PingOne password.
- For **Drivers**, choose **Amazon Redshift (com.amazon.redshift.jdbc.Driver)**.
- For **URL**, enter **`jdbc:redshift:iam://`*`your-cluster-identifier`*`:`*`your-cluster-region`*`/`*`your-database-name`***.

e. Choose **Extended Properties** and do one of the following:

- For **login_url**, enter ***`your-ping-sso-login-url`***. This value specifies to the URL to use single sign-on as the authentication to log in.
- For Ping Identity, for **plugin_name**, enter **`com.amazon.redshift.plugin.PingCredentialsProvider`**. This value specifies to the driver to use Ping Identity single sign-on as the authentication method.

- For Ping Identity with single sign-on, for **plugin_name**, enter **`com.amazon.redshift.plugin.BrowserSamlCredentialsProvider`**. This value specifies to the driver to use Ping Identity PingOne with single sign-on as the authentication method.

ODBC

To set up ODBC for authentication to Ping Identity

- Configure your database client to connect to your cluster through ODBC using Ping Identity PingOne single sign-on.

  Amazon Redshift provides ODBC drivers for Linux, Windows, and macOS operating systems. Before you install an ODBC driver, determine whether your SQL client tool is 32-bit or 64-bit. Install the ODBC driver that matches the requirements of your SQL client tool.

  On Windows, in the **Amazon Redshift ODBC Driver DSN Setup** page, under **Connection Settings**, enter the following information:

  - For **Data Source Name**, enter ***`your-DSN`***. This specifies the data source name used as the ODBC profile name.
  - For **Auth type**, do one of the following:
    - For Ping Identity configuration, choose **Identity Provider: Ping Federate**. This is the authentication method that the ODBC driver uses to authenticate using Ping Identity single sign-on.
    - For Ping Identity with single sign-on configuration, choose **Identity Provider: Browser SAML**. This is the authentication method that the ODBC driver uses to authenticate using Ping Identity with single sign-on.
  - For **Cluster ID**, enter ***`your-cluster-identifier`***.
  - For **Region**, enter ***`your-cluster-region`***.
  - For **Database**, enter ***`your-database-name`***.
  - For **User**, enter ***`your-ping-username`***. This is the user name for the PingOne account that you are using for single sign-on that has permission to the cluster that you're trying to authenticate using. Use this only for **Auth type** is **Identity Provider: PingFederate**.
  - For **Password**, enter ***`your-ping-password`***. Use this only for **Auth type** is **Identity Provider: PingFederate**.

- For **Listen Port**, enter ***your-listen-port***. This is the port that local server is listening to. The default is 7890. This applies only to the Browser SAML plugin.
- For **Response Timeout**, enter ***the-number-of-seconds***. This is the number of seconds to wait before timing out when the IdP server sends back a response. The minimum number of seconds must be 10. If establishing the connection takes longer than this threshold, then the connection is aborted. This applies only to the Browser SAML plugin.
- For **Login URL**, enter ***your-login-url***. This applies only to the Browser SAML plugin.

On macOS and Linux, edit the `odbc.ini` file as follows:

> **Note**
>
> All entries are case-insensitive.

- For **clusterid**, enter ***your-cluster-identifier***. This is the name of the created Amazon Redshift cluster.
- For **region**, enter ***your-cluster-region***. This is the AWS Region of the created Amazon Redshift cluster.
- For **database**, enter ***your-database-name***. This is the name of the database that you're trying to access on the Amazon Redshift cluster.
- For **locale**, enter **en-us**. This is the language that error messages display in.
- For **iam**, enter **1**. This value specifies to the driver to authenticate using IAM credentials.
- For **plugin_name**, do one of the following:
  - For Ping Identity configuration, enter **BrowserSAML**. This is the authentication method that the ODBC driver uses to authenticate to Ping Identity.
  - For Ping Identity with single sign-on configuration, enter **Ping**. This is the authentication method that the ODBC driver uses to authenticate using Ping Identity with single sign-on.
- For **uid**, enter ***your-ping-username***. This is the user name of the Microsoft Azure account you are using for single sign-on that has permission to the cluster you are trying to authenticate against. Use this only for **plugin_name** is **Ping**.
- For **pwd**, enter ***your-ping-password***. Use this only for **plugin_name** is **Ping**.

- For **login_url**, enter ***your-login-url***. This is the Initiate single sign-on URL that returns the SAML Response. This applies only to the Browser SAML plugin.
- For **idp_response_timeout**, enter ***the-number-of-seconds***. This is the specified period of time in seconds to wait for response from PingOne Identity. This applies only to the Browser SAML plugin.
- For **listen_port**, enter ***your-listen-port***. This is the port that local server is listening to. The default is 7890. This applies only to the Browser SAML plugin.

On macOS and Linux, also edit the profile settings to add the following exports.

```
export ODBCINI=/opt/amazon/redshift/Setup/odbc.ini
```

```
export ODBCINSTINI=/opt/amazon/redshift/Setup/odbcinst.ini
```

Okta

You can use Okta as an identity provider (IdP) to access your Amazon Redshift cluster. This tutorial shows you how you can use Okta as an identity provider (IdP) to access your Amazon Redshift cluster.

Step 1: Set up Okta and your AWS account to trust each other

The following procedure describes how to set up a trust relationship.

To set up Okta and your AWS account to trust each other

1. Create or use an existing Amazon Redshift cluster for your Okta users to connect to. To configure the connection, certain properties of this cluster are needed, such as the cluster identifier. For more information, see Creating a Cluster.
2. Add Amazon Redshift as a new application on the Okta portal. For detailed steps, see the Okta documentation.
   - Choose **Add Application**.
   - Under **Add Application**, choose **Create New App**.
   - On the **Create a New Add Application Integration** page, for **Platform**, choose **Web**.
   - For **Sign on method**, choose **SAML v2.0**.

- On the **General Settings** page, for **App name**, enter ***your-redshift-saml-sso-name***. This is the name of your application.
- On the **SAML Settings** page, for **Single sign on URL**, enter ***your-redshift-local-host-url***. This is the local host and port that the SAML assertion redirects to, for example `http://localhost:7890/redshift/`.

3. Use the **Single sign on URL** value as the **Recipient URL** and **Destination URL**.
4. For **Signing**, choose **Sign Assertion**.
5. For **Audience URI (SP Entity ID)**, enter **`urn:amazon:webservices`** for the claims, as shown in the following table.
6. In the **Advanced Settings** section, for **SAML Issuer ID**, enter ***your-Identity-Provider-Issuer-ID***, which you can find in the **View Setup Instructions** section.
7. In the **Attribute Statements** section, create the claims as shown in the following table.

Claim name	Value
https://aws.amazon.com/SAML/Attributes/Role	arn:aws:iam::*123456789012* :role/*Okta*,arn:aws:iam::*123456789012* :saml-provider/*Okta*
https://aws.amazon.com/SAML/Attributes/RoleSessionName	user.email
https://redshift.amazon.com/SAML/Attributes/AutoCreate	"true"
https://redshift.amazon.com/SAML/Attributes/DbUser	user.email

8. In the **App Embed Link** section, find the URL that you can use as the login URL for the Browser SAML plugin.
9. Create an IAM SAML identity provider on the IAM console. The metadata document that you provide is the federation metadata XML file that you saved when you set up Okta. For detailed steps, see Creating and Managing an IAM Identity Provider (Console) in the *IAM User Guide*.
10. Create an IAM role for SAML 2.0 federation on the IAM console. For detailed steps, see Creating a Role for SAML in the *IAM User Guide*.

11. Create an IAM policy that you can attach to the IAM role that you created for SAML 2.0 federation on the IAM console. For detailed steps, see Creating IAM Policies (Console) in the *IAM User Guide*. For an Azure AD example, see Setting up JDBC or ODBC single sign-on authentication.

Step 2: Set up JDBC or ODBC for authentication to Okta

JDBC

To set up JDBC for authentication to Okta

- Configure your database client to connect to your cluster through JDBC using Okta single sign-on.

  You can use any client that uses a JDBC driver to connect using Okta single sign-on or use a language like Java to connect using a script. For installation and configuration information, see Configuring a connection for JDBC driver version 2.x for Amazon Redshift.

  For example, you can use SQLWorkbench/J as the client. When you configure SQLWorkbench/J, the URL of your database uses the following format.

  ```
  jdbc:redshift:iam://cluster-identifier:us-west-1/dev
  ```

  If you use SQLWorkbench/J as the client, take the following steps:

  a. Start SQL Workbench/J. In the **Select Connection Profile** page, add a **Profile Group**, for example **`Okta`**.

  b. For **Connection Profile**, enter ***`your-connection-profile-name`***, for example **`Okta`**.

  c. Choose **Manage Drivers**, and choose **Amazon Redshift**. Choose the **Open Folder** icon next to **Library**, then choose the appropriate JDBC .jar file.

  d. On the **Select Connection Profile** page, add information to the connection profile as follows:

  - For **User**, enter your Okta user name. This is the user name of the Okta account that you are using for single sign-on that has permission to the cluster that you are trying to authenticate using.
  - For **Password**, enter your Okta password.

- For **Drivers**, choose **Amazon Redshift (com.amazon.redshift.jdbc.Driver)**.
- For **URL**, enter **`jdbc:redshift:iam://`*`your-cluster-identifier`*`:`*`your-cluster-region`*`/`*`your-database-name`**.

e. Choose **Extended Properties** and do one of the following:

- For **login_url**, enter ***`your-okta-sso-login-url`***. This value specifies to the URL to use single sign-on as the authentication to log in to Okta.
- For Okta single sign-on, for **plugin_name**, enter **`com.amazon.redshift.plugin.OktaCredentialsProvider`**. This value specifies to the driver to use Okta single sign-on as the authentication method.
- For Okta single sign-on with MFA, for **plugin_name**, enter **`com.amazon.redshift.plugin.BrowserSamlCredentialsProvider`**. This value specifies to the driver to use Okta single sign-on with MFA as the authentication method.

ODBC

To set up ODBC for authentication to Okta

- Configure your database client to connect to your cluster through ODBC using Okta single sign-on.

  Amazon Redshift provides ODBC drivers for Linux, Windows, and macOS operating systems. Before you install an ODBC driver, determine whether your SQL client tool is 32-bit or 64-bit. Install the ODBC driver that matches the requirements of your SQL client tool.

  On Windows, in the **Amazon Redshift ODBC Driver DSN Setup** page, under **Connection Settings**, enter the following information:

  - For **Data Source Name**, enter ***`your-DSN`***. This specifies the data source name used as the ODBC profile name.
  - For **Auth type**, do one of the following:
    - For Okta single sign-on configuration, choose **`Identity Provider: Okta`**. This is the authentication method that the ODBC driver uses to authenticate using Okta single sign-on.

  - For Okta single sign-on with MFA configuration, choose **`Identity Provider: Browser SAML`**. This is the authentication method that the ODBC driver uses to authenticate using Okta single sign-on with MFA.
- For **Cluster ID**, enter ***`your-cluster-identifier`***.
- For **Region**, enter ***`your-cluster-region`***.
- For **Database**, enter ***`your-database-name`***.
- For **User**, enter ***`your-okta-username`***. This is the user name for the Okta account that you are using for single sign-on that has permission to the cluster that you're trying to authenticate using. Use this only for **Auth type** is **Identity Provider: Okta**.
- For **Password**, enter ***`your-okta-password`***. Use this only for **Auth type** is **Identity Provider: Okta**.

On macOS and Linux, edit the odbc`.ini` file as follows:

> **Note**
>
> All entries are case-insensitive.

- For **clusterid**, enter ***`your-cluster-identifier`***. This is the name of the created Amazon Redshift cluster.
- For **region**, enter ***`your-cluster-region`***. This is the AWS Region of the created Amazon Redshift cluster.
- For **database**, enter ***`your-database-name`***. This is the name of the database that you're trying to access on the Amazon Redshift cluster.
- For **locale**, enter **`en-us`**. This is the language that error messages display in.
- For **iam**, enter **`1`**. This value specifies to the driver to authenticate using IAM credentials.
- For **plugin_name**, do one of the following:
  - For Okta single sign-on with MFA configuration, enter **`BrowserSAML`**. This is the authentication method that the ODBC driver uses to authenticate to Okta single sign-on with MFA.
  - For Okta single sign-on configuration, enter **`Okta`**. This is the authentication method that the ODBC driver uses to authenticate using Okta single sign-on.

- For **uid**, enter ***your-okta-username***. This is the user name of the Okta account you are using for single sign-on that has permission to the cluster you are trying to authenticate against. Use this only for **plugin_name** is **Okta**.
- For **pwd**, enter ***your-okta-password***. Use this only for **plugin_name** is **Okta**.
- For **login_url**, enter ***your-login-url***. This is the Initiate single sign-on URL that returns the SAML Response. This applies only to the Browser SAML plugin.
- For **idp_response_timeout**, enter ***the-number-of-seconds***. This is the specified period of time in seconds to wait for response from PingOne. This applies only to the Browser SAML plugin.
- For **listen_port**, enter ***your-listen-port***. This is the port that local server is listening to. The default is 7890. This applies only to the Browser SAML plugin.

On macOS and Linux, also edit the profile settings to add the following exports.

```
export ODBCINI=/opt/amazon/redshift/Setup/odbc.ini
```

```
export ODBCINSTINI=/opt/amazon/redshift/Setup/odbcinst.ini
```

Authorizing Amazon Redshift to access AWS services on your behalf

Some Amazon Redshift features require Amazon Redshift to access other AWS services on your behalf. For example, the COPY and UNLOAD commands can load or unload data into your Amazon Redshift cluster using an Amazon S3 bucket. The CREATE EXTERNAL FUNCTION command can invoke an AWS Lambda function using a scalar Lambda user-defined function (UDF). Amazon Redshift Spectrum can use a data catalog in Amazon Athena or AWS Glue. For your Amazon Redshift clusters to act on your behalf, you supply security credentials to your clusters. The preferred method to supply security credentials is to specify an AWS Identity and Access Management (IAM) role. For COPY and UNLOAD, you can provide temporary credentials.

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
IAM	(Recommended) Use console credentials as temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Login for AWS local development in the *AWS Command Line Interface User Guide*. • For AWS SDKs, see Login for AWS local development in the *AWS SDKs and Tools Reference Guide*.
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.

Which user needs programmatic access?	To	By
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

Following, find out how to create an IAM role with the appropriate permissions to access other AWS services. You also need to associate the role with your cluster and specify the Amazon Resource Name (ARN) of the role when you run the Amazon Redshift command. For more information, see Authorizing COPY, UNLOAD, CREATE EXTERNAL FUNCTION, and CREATE EXTERNAL SCHEMA operations using IAM roles.

In addition, a superuser can grant the ASSUMEROLE privilege to specific users and groups to provide access to a role for COPY and UNLOAD operations. For information, see GRANT in the *Amazon Redshift Database Developer Guide*.

Creating an IAM role to allow your Amazon Redshift cluster to access AWS services

Creating an IAM role to allow your Amazon Redshift cluster to access AWS services

To create an IAM role to permit your Amazon Redshift cluster to communicate with other AWS services on your behalf, take the following steps. The values used in this section are examples, you can choose values based on your needs.

To create an IAM role to allow Amazon Redshift to access AWS services

1. Open the IAM console.
2. In the navigation pane, choose **Roles**.
3. Choose **Create role**.
4. Choose **AWS service**, and then choose **Redshift**.
5. Under **Select your use case**, choose **Redshift - Customizable** and then choose **Next: Permissions**. The **Attach permissions policy** page appears.
6. For access to Amazon S3 using COPY, as an example, you can use **`AmazonS3ReadOnlyAccess`** and append. For access to Amazon S3 using COPY or UNLOAD, we suggest that you can create managed policies that restrict access to the desired bucket and prefix accordingly. For both read and write operations, we recommend enforcing the least privileges and restricting to only the Amazon S3 buckets and key prefixes that Amazon Redshift requires.

   For access to invoke Lambda functions for the CREATE EXTERNAL FUNCTION command, add **`AWSLambdaRole`**.

   For Redshift Spectrum, in addition to Amazon S3 access, add **`AWSGlueConsoleFullAccess`** or **`AmazonAthenaFullAccess`**.

   Choose **Next: Tags**.
7. The **Add tags** page appears. You can optionally add tags. Choose **Next: Review**.
8. For **Role name**, type a name for your role, for example **`RedshiftCopyUnload`**. Choose **Create role**.
9. The new role is available to all users on clusters that use the role. To restrict access to only specific users on specific clusters, or to clusters in specific regions, edit the trust relationship for the role. For more information, see Restricting access to IAM roles.

10. Associate the role with your cluster. You can associate an IAM role with a cluster when you create the cluster, or you add the role to an existing cluster. For more information, see Associating IAM roles with clusters.

    > **Note**
    >
    > To restrict access to specific data, use an IAM role that grants the least privileges required.

Restricting access to IAM roles

By default, IAM roles that are available to an Amazon Redshift cluster are available to all users on that cluster. You can choose to restrict IAM roles to specific Amazon Redshift database users on specific clusters or to specific regions.

To permit only specific database users to use an IAM role, take the following steps.

To identify specific database users with access to an IAM role

1. Identify the Amazon Resource Name (ARN) for the database users in your Amazon Redshift cluster. The ARN for a database user is in the format: `arn:aws:redshift:region:account-id:dbuser:cluster-name/user-name`.

   For Amazon Redshift Serverless use the following ARN format. `arn:aws:redshift:region:account-id:dbuser:serverless-account-id-workgroup-id/user-name`
2. Open the IAM console.
3. In the navigation pane, choose **Roles**.
4. Choose the IAM role that you want to restrict to specific Amazon Redshift database users.
5. Choose the **Trust Relationships** tab, and then choose **Edit Trust Relationship**. A new IAM role that allows Amazon Redshift to access other AWS services on your behalf has a trust relationship as follows:

   JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

6. Add a condition to the `sts:AssumeRole` action section of the trust relationship that limits the `sts:ExternalId` field to values that you specify. Include an ARN for each database user that you want to grant access to the role. The external ID can be any unique string.

   For example, the following trust relationship specifies that only database users `user1` and `user2` on cluster `my-cluster` in region `us-west-2` have permission to use this IAM role.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "Service": "redshift.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "sts:ExternalId": [
          "arn:aws:redshift:us-west-2:123456789012:dbuser:my-cluster/user1",
          "arn:aws:redshift:us-west-2:123456789012:dbuser:my-cluster/user2"
        ]
      }
    }
  }]
}
```

7. Choose **Update Trust Policy**.

Restricting an IAM role to an AWS Region

You can restrict an IAM role to only be accessible in a certain AWS Region. By default, IAM roles for Amazon Redshift are not restricted to any single region.

To restrict use of an IAM role by region, take the following steps.

To identify permitted regions for an IAM role

1. Open the IAM console at https://console.aws.amazon.com/.
2. In the navigation pane, choose **Roles**.
3. Choose the role that you want to modify with specific regions.
4. Choose the **Trust Relationships** tab and then choose **Edit Trust Relationship**. A new IAM role that allows Amazon Redshift to access other AWS services on your behalf has a trust relationship as follows:

   JSON

   ```
   {
     "Version":"2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "redshift.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

5. Modify the `Service` list for the `Principal` with the list of the specific regions that you want to permit use of the role for. Each region in the `Service` list must be in the following format: `redshift.`*`region`*`.amazonaws.com`.

   For example, the following edited trust relationship permits the use of the IAM role in the `us-east-1` and `us-west-2` regions only.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "redshift.us-east-1.amazonaws.com",
          "redshift.us-west-2.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

6. Choose **Update Trust Policy**

Chaining IAM roles in Amazon Redshift

When you attach a role to your cluster, your cluster can assume that role to access Amazon S3, Amazon Athena, AWS Glue, and AWS Lambda on your behalf. If a role attached to your cluster doesn't have access to the necessary resources, you can chain another role, possibly belonging to another account. Your cluster then temporarily assumes the chained role to access the data. You can also grant cross-account access by chaining roles. Each role in the chain assumes the next role in the chain, until the cluster assumes the role at the end of chain. The maximum number of IAM roles that you can associate is subject to a quota. For more information, see the quota "Cluster IAM roles for Amazon Redshift to access other AWS services" in Quotas for Amazon Redshift objects.

> **Note**
>
> You must specify the IAM roles in order for the chain to work correctly.

For example, suppose Company A wants to access data in an Amazon S3 bucket that belongs to Company B. Company A creates an AWS service role for Amazon Redshift named `RoleA` and attaches it to their cluster. Company B creates a role named `RoleB` that's authorized to access the

data in the Company B bucket. To access the data in the Company B bucket, Company A runs a COPY command using an `iam_role` parameter that chains `RoleA` and `RoleB`. For the duration of the COPY operation, `RoleA` temporarily assumes `RoleB` to access the Amazon S3 bucket.

To chain roles, you establish a trust relationship between the roles. A role that assumes another role (for example, `RoleA`) must have a permissions policy that allows it to assume the next chained role (for example, `RoleB`). In turn, the role that passes permissions (`RoleB`) must have a trust policy that allows it to pass its permissions to the previous chained role (`RoleA`). For more information, see Using IAM roles in the IAM User Guide.

The first role in the chain must be a role attached to the cluster. The first role, and each subsequent role that assumes the next role in the chain, must have a policy that includes a specific statement. This statement has the `Allow` effect on the `sts:AssumeRole` action and the Amazon Resource Name (ARN) of the next role in a `Resource` element. In our example, `RoleA` has the following permission policy that allows it to assume `RoleB`, owned by AWS account `210987654321`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1487639602000",
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/RoleB"
       }
    ]
}
```

A role that passes to another role must establish a trust relationship with the role that assumes the role or with the AWS account that owns the role. In our example, `RoleB` has the following trust policy to establish a trust relationship with `RoleA`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/RoleA"
      }
    }
  ]
}
```

The following trust policy establishes a trust relationship with the owner of `RoleA`, AWS account `123456789012`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      }
    }
  ]
}
```

Note

To restrict role chaining authorization to specific users, define a condition. For more information, see Restricting access to IAM roles.

When you run an UNLOAD, COPY, CREATE EXTERNAL FUNCTION, or CREATE EXTERNAL SCHEMA command, you chain roles by including a comma-separated list of role ARNs in the `iam_role` parameter. The following shows the syntax for chaining roles in the `iam_role` parameter.

```
unload ('select * from venue limit 10')
to 's3://acmedata/redshift/venue_pipe_'
IAM_ROLE 'arn:aws:iam::<aws-account-id-1>:role/<role-name-1>[,arn:aws:iam::<aws-account-id-2>:role/<role-name-2>][,...]';
```

Note

The entire role chain is enclosed in single quotes and must not contain spaces.

In the following examples, `RoleA` is attached to the cluster belonging to AWS account `123456789012`. `RoleB`, which belongs to account `210987654321`, has permission to access the bucket named `s3://companyb/redshift/`. The following example chains `RoleA` and `RoleB` to UNLOAD data to the s3://companyb/redshift/ bucket.

```
unload ('select * from venue limit 10')
to 's3://companyb/redshift/venue_pipe_'
iam_role 'arn:aws:iam::123456789012:role/RoleA,arn:aws:iam::210987654321:role/RoleB';
```

The following example uses a COPY command to load the data that was unloaded in the previous example.

```
copy venue
from 's3://companyb/redshift/venue_pipe_'
iam_role 'arn:aws:iam::123456789012:role/RoleA,arn:aws:iam::210987654321:role/RoleB';
```

In the following example, CREATE EXTERNAL SCHEMA uses chained roles to assume the role `RoleB`.

```
create external schema spectrumexample from data catalog
database 'exampledb' region 'us-west-2'
iam_role 'arn:aws:iam::123456789012:role/RoleA,arn:aws:iam::210987654321:role/RoleB';
```

In the following example, CREATE EXTERNAL FUNCTION uses chained roles to assume the role `RoleB`.

```
create external function lambda_example(varchar)
returns varchar
volatile
lambda 'exampleLambdaFunction'
iam_role 'arn:aws:iam::123456789012:role/RoleA,arn:aws:iam::210987654321:role/RoleB';
```

Authorizing COPY, UNLOAD, CREATE EXTERNAL FUNCTION, and CREATE EXTERNAL SCHEMA operations using IAM roles

You can use the COPY command to load (or import) data into Amazon Redshift and the UNLOAD command to unload (or export) data from Amazon Redshift. You can use the CREATE EXTERNAL FUNCTION command to create user-defined functions that invoke functions from AWS Lambda.

When you use Amazon Redshift Spectrum, you use the CREATE EXTERNAL SCHEMA command to specify the location of an Amazon S3 bucket that contains your data. When you run the COPY, UNLOAD, or CREATE EXTERNAL SCHEMA commands, you provide security credentials. These credentials authorize your Amazon Redshift cluster to read or write data to and from your target destination, such as an Amazon S3 bucket.

When you run the CREATE EXTERNAL FUNCTION, you provide security credentials using the IAM role parameter. These credentials authorize your Amazon Redshift cluster to invoke Lambda functions from AWS Lambda. The preferred method to supply security credentials is to specify an AWS Identity and Access Management (IAM) role. For COPY and UNLOAD, you can provide temporary credentials. For information about creating an IAM role, see Authorizing Amazon Redshift to access AWS services on your behalf.

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
IAM	(Recommended) Use console credentials as temporary credentials to sign programmatic requests to the	Following the instructions for the interface that you want to use.

Which user needs programmatic access?	To	By
	AWS CLI, AWS SDKs, or AWS APIs.	• For the AWS CLI, see Login for AWS local development in the *AWS Command Line Interface User Guide*. • For AWS SDKs, see Login for AWS local development in the *AWS SDKs and Tools Reference Guide*.
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.

Which user needs programmatic access?	To	By
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

The steps for using an IAM role are as follows:

- Create an IAM role for use with your Amazon Redshift cluster.
- Associate the IAM role with the cluster.
- Include the IAM role's ARN when you call the COPY, UNLOAD, CREATE EXTERNAL SCHEMA, or CREATE EXTERNAL FUNCTION command.

Associating IAM roles with clusters

After you have created an IAM role that authorizes Amazon Redshift to access other AWS services for you, you must associate that role with an Amazon Redshift cluster. You must do this before you can use the role to load or unload data.

Permissions required to associate an IAM role with a cluster

To associate an IAM role with a cluster, a user must have `iam:PassRole` permission for that IAM role. This permission allows an administrator to restrict which IAM roles a user can associate with Amazon Redshift clusters. As a best practice, we recommend attaching permissions policies to an IAM role and then assigning it to users and groups as needed. For more information, see Identity and access management in Amazon Redshift.

The following example shows an IAM policy that can be attached to a user that allows the user to take these actions:

- Get the details for all Amazon Redshift clusters owned by that user's account.
- Associate any of three IAM roles with either of two Amazon Redshift clusters.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "redshift:DescribeClusters",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                 "redshift:ModifyClusterIamRoles",
                 "redshift:CreateCluster"
            ],
            "Resource": [
                 "arn:aws:redshift:us-east-1:123456789012:cluster:my-redshift-
cluster",
                 "arn:aws:redshift:us-east-1:123456789012:cluster:my-second-
redshift-cluster"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
```

```
                "arn:aws:iam::123456789012:role/MyRedshiftRole",
                "arn:aws:iam::123456789012:role/SecondRedshiftRole",
                "arn:aws:iam::123456789012:role/ThirdRedshiftRole"
            ]
        }
    ]
}
```

After a user has the appropriate permissions, that user can associate an IAM role with an Amazon Redshift cluster. The IAM role is then ready to use with the COPY or UNLOAD command or other Amazon Redshift commands.

For more information on IAM policies, see Overview of IAM policies in the *IAM User Guide*.

Managing IAM role association with a cluster

You can associate an IAM role with an Amazon Redshift cluster when you create the cluster. Or you can modify an existing cluster and add or remove one or more IAM role associations.

Be aware of the following:

- The maximum number of IAM roles that you can associate is subject to a quota.
- An IAM role can be associated with multiple Amazon Redshift clusters.
- An IAM role can be associated with an Amazon Redshift cluster only if both the IAM role and the cluster are owned by the same AWS account.

You can manage IAM role associations for a cluster with the console by using the following procedure.

To manage IAM role associations

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the cluster that you want to update.
3. For **Actions**, choose **Manage IAM roles** to display the current list IAM roles associated with the cluster.
4. On the **Manage IAM roles** page, choose the available IAM roles to add, and then choose **Add IAM role**.

5. Choose **Done** to save your changes.

You can manage IAM role associations for a cluster with the AWS CLI by using the following approaches.

To associate an IAM role with a cluster when the cluster is created, specify the Amazon Resource Name (ARN) of the IAM role for the `--iam-role-arns` parameter of the `create-cluster` command. The maximum number of IAM roles that you can add when calling the `create-cluster` command is subject to a quota.

Associating and disassociating IAM roles with Amazon Redshift clusters is an asynchronous process. You can get the status of all IAM role cluster associations by calling the `describe-clusters` command.

The following example associates two IAM roles with the newly created cluster named `my-redshift-cluster`.

```
aws redshift create-cluster \
    --cluster-identifier "my-redshift-cluster" \
    --node-type "ra3.4xlarge" \
    --number-of-nodes 16 \
    --iam-role-arns "arn:aws:iam::123456789012:role/RedshiftCopyUnload" \
                    "arn:aws:iam::123456789012:role/SecondRedshiftRole"
```

To associate an IAM role with an existing Amazon Redshift cluster, specify the Amazon Resource Name (ARN) of the IAM role for the `--add-iam-roles` parameter of the `modify-cluster-iam-roles` command. The maximum number of IAM roles that you can add when calling the `modify-cluster-iam-roles` command is subject to a quota.

The following example associates an IAM role with an existing cluster named `my-redshift-cluster`.

```
aws redshift modify-cluster-iam-roles \
    --cluster-identifier "my-redshift-cluster" \
    --add-iam-roles "arn:aws:iam::123456789012:role/RedshiftCopyUnload"
```

To disassociate an IAM role from a cluster, specify the ARN of the IAM role for the `--remove-iam-roles` parameter of the `modify-cluster-iam-roles` command. `modify-cluster-iam-roles` The maximum number of IAM roles that you can remove when calling the `modify-cluster-iam-roles` command is subject to a quota.

The following example removes the association for an IAM role for the 123456789012 AWS account from a cluster named my-redshift-cluster.

```
aws redshift modify-cluster-iam-roles \
    --cluster-identifier "my-redshift-cluster" \
    --remove-iam-roles "arn:aws:iam::123456789012:role/RedshiftCopyUnload"
```

Listing IAM role associations for a cluster using the AWS CLI

To list all of the IAM roles that are associated with an Amazon Redshift cluster, and the status of the IAM role association, call the describe-clusters command. The ARN for each IAM role associated with the cluster is returned in the IamRoles list as shown in the following example output.

Roles that have been associated with the cluster show a status of in-sync. Roles that are in the process of being associated with the cluster show a status of adding. Roles that are being disassociated from the cluster show a status of removing.

```
{
    "Clusters": [
        {
            "ClusterIdentifier": "my-redshift-cluster",
            "NodeType": "ra3.4xlarge",
            "NumberOfNodes": 16,
            "IamRoles": [
                {
                    "IamRoleArn": "arn:aws:iam::123456789012:role/MyRedshiftRole",
                    "IamRoleApplyStatus": "in-sync"
                },
                {
                    "IamRoleArn": "arn:aws:iam::123456789012:role/SecondRedshiftRole",
                    "IamRoleApplyStatus": "in-sync"
                }
            ],
            ...
        },
        {
            "ClusterIdentifier": "my-second-redshift-cluster",
            "NodeType": "ra3.4xlarge",
            "NumberOfNodes": 10,
            "IamRoles": [
```

```
                {
                    "IamRoleArn": "arn:aws:iam::123456789012:role/MyRedshiftRole",
                    "IamRoleApplyStatus": "in-sync"
                },
                {
                    "IamRoleArn": "arn:aws:iam::123456789012:role/SecondRedshiftRole",
                    "IamRoleApplyStatus": "in-sync"
                },
                {
                    "IamRoleArn": "arn:aws:iam::123456789012:role/ThirdRedshiftRole",
                    "IamRoleApplyStatus": "in-sync"
                }
            ],
            ...
        }
    ]
}
```

For more information on using the AWS CLI, see *AWS CLI User Guide*.

Creating an IAM role as default for Amazon Redshift

When you create IAM roles through the Redshift console, Amazon Redshift programmatically creates the roles in your AWS account and automatically attaches existing AWS managed policies to them. This approach means that you can stay within the Redshift console and don't have to switch to the IAM console for role creation. For more granular control of permissions for an existing IAM role that was created in the Amazon Redshift console, you can attach a customized managed policy to the IAM role.

IAM roles created in the console

When you use the Amazon Redshift console to create IAM roles, Amazon Redshift tracks all IAM roles created through the console. Amazon Redshift preselects the most recent default IAM role for creating all new clusters and restoring clusters from snapshots.

You can create an IAM role through the console that has a policy with permissions to run SQL commands. These commands include COPY, UNLOAD, CREATE EXTERNAL FUNCTION, CREATE EXTERNAL TABLE, CREATE EXTERNAL SCHEMA, CREATE MODEL, or CREATE LIBRARY. Optionally, you can get more granular control of user access to your AWS resources by creating and attaching custom policies to the IAM role.

When you created an IAM role and set it as the default for the cluster using console, you don't have to provide the IAM role's Amazon Resource Name (ARN) to perform authentication and authorization.

The IAM role that you create through the console for your cluster has the `AmazonRedshiftAllCommandsFullAccess` managed policy automatically attached. This IAM role allows Amazon Redshift to copy, unload, query, and analyze data for AWS resources in your IAM account. The managed policy provides access to COPY, UNLOAD, CREATE EXTERNAL FUNCTION, CREATE EXTERNAL SCHEMA, CREATE MODEL, and CREATE LIBRARY operations. The policy also grants permissions to run SELECT statements for related AWS services, such as Amazon S3, Amazon CloudWatch Logs, Amazon SageMaker AI, and AWS Glue.

The CREATE EXTERNAL FUNCTION, CREATE EXTERNAL SCHEMA, CREATE MODEL, and CREATE LIBRARY commands have a `default` keyword. For this keyword for these commands, Amazon Redshift uses the IAM role that is set as the default and associated with the cluster when the command runs. You can run the DEFAULT_IAM_ROLE command to check the current default IAM role that is attached to the cluster.

To control access privileges of the IAM role created and set as default for your Redshift cluster, use the ASSUMEROLE privilege. This access control applies to database users and groups when they run commands such as the ones listed preceding. After you grant the ASSUMEROLE privilege to a user or group for the IAM role, the user or group can assume that role when running these commands. By using the ASSUMEROLE privilege, you can grant access to the appropriate commands as required.

Using the Amazon Redshift console, you can do the following:

- Creating an IAM role as the default
- Removing IAM roles from your cluster
- Associating IAM roles with your cluster
- Setting an IAM role as the default
- Making an IAM role no longer default for your cluster

Permissions of the AmazonRedshiftAllCommandsFullAccess managed policy

The following example shows the permissions in the `AmazonRedshiftAllCommandsFullAccess` managed policy that allow certain actions for the IAM role that is set as default for your cluster. The IAM role with permission policies attached authorizes what a user or group can and can't do.

Given these permissions, you can run the COPY command from Amazon S3, run UNLOAD, and use the CREATE MODEL command.

```
{
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetBucketAcl",
                "s3:GetBucketCors",
                "s3:GetEncryptionConfiguration",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:ListMultipartUploadParts",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject",
                "s3:PutBucketAcl",
                "s3:PutBucketCors",
                "s3:DeleteObject",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket"
            ],
            "Resource": [
                "arn:aws:s3:::redshift-downloads",
                "arn:aws:s3:::redshift-downloads/*",
                "arn:aws:s3:::*redshift*",
                "arn:aws:s3:::*redshift*/*"
            ]
}
```

The following example shows the permissions in the `AmazonRedshiftAllCommandsFullAccess` managed policy that allow certain actions for the IAM role that is set as default for the cluster. The IAM role with permission policies attached authorizes what a user or group can and can't do. Given the following permissions, you can run the CREATE EXTERNAL FUNCTION command.

```
{
    "Action": [
        "lambda:InvokeFunction"
    ],
    "Resource": "arn:aws:lambda:*:*:function:*redshift*"
}
```

The following example shows the permissions in the AmazonRedshiftAllCommandsFullAccess managed policy that allow certain actions for the IAM role that is set as default for the cluster. The IAM role with permission policies attached authorizes what a user or group can and can't do. Given the following permissions, you can run the CREATE EXTERNAL SCHEMA and CREATE EXTERNAL TABLE commands needed for Amazon Redshift Spectrum.

```
{
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:DeleteDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:CreateTable",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition"
            ],
            "Resource": [
                "arn:aws:glue:*:*:table/*redshift*/*",
                "arn:aws:glue:*:*:catalog",
                "arn:aws:glue:*:*:database/*redshift*"
            ]
}
```

The following example shows the permissions in the AmazonRedshiftAllCommandsFullAccess managed policy that allow certain actions for the IAM role set as default for the cluster. The IAM role with permission policies attached authorizes what a user or group can and can't do. Given the following permissions, you can run the CREATE EXTERNAL SCHEMA command using federated queries.

```
{
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": [
                "arn:aws:secretsmanager:*:*:secret:*Redshift*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword",
                "secretsmanager:ListSecrets"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:ResourceTag/Redshift": "true"
                }
            }
},
```

Managing IAM roles created for a cluster using the console

To create, modify, and remove IAM roles created from the Amazon Redshift console, use the **Clusters** section in the console.

Creating an IAM role as the default

On the console, you can create an IAM role for your cluster that has the `AmazonRedshiftAllCommandsFullAccess` policy automatically attached. The new IAM role that you create allows Amazon Redshift to copy, load, query, and analyze data from Amazon resources in your IAM account.

There can only be one IAM role set as the default for the cluster. If you create another IAM role as the cluster default when an existing IAM role is currently assigned as the default, the new IAM role replaces the other one as default.

To create a new cluster and an IAM role set as the default for the new cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**. The clusters for your account in the current AWS Region are listed. A subset of properties of each cluster is displayed in columns in the list.
3. Choose **Create cluster** to create a cluster.
4. Follow the instructions on the console page to enter the properties for **Cluster configuration**. For more information about this step, see Creating a cluster.
5. (Optional) Choose **Load sample data** to load the sample data set to your Amazon Redshift cluster to start using the query editor to query data.

   If you are behind a firewall, the database port must be an open port that accepts inbound connections.
6. Follow the instructions on the console page to enter properties for **Database configurations**.
7. Under **Cluster permissions**, from **Manage IAM roles**, choose **Create IAM role**.
8. Specify an Amazon S3 bucket for the IAM role to access by choosing one of the following methods:

   - Choose **No additional Amazon S3 bucket** to create the IAM role without specifying specific Amazon S3 buckets.
   - Choose **Any Amazon S3 bucket** to allow users that have access to your Amazon Redshift cluster to also access any Amazon S3 bucket and its contents in your AWS account.
   - Choose **Specific Amazon S3 buckets** to specify one or more Amazon S3 buckets that the IAM role being created has permission to access. Then choose one or more Amazon S3 buckets from the table.
9. Choose **Create IAM role as default**. Amazon Redshift automatically creates and sets the IAM role as the default for your cluster.
10. Choose **Create cluster** to create the cluster. The cluster might take several minutes to be ready to use.

Removing IAM roles from your cluster

You can remove one or more IAM roles from your cluster.

To remove IAM roles from your cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**. The clusters for your account in the current AWS Region are listed. A subset of properties of each cluster is displayed in columns in the list.
3. Choose the cluster that you want to remove the IAM role from.
4. Under **Cluster permissions**, choose one or more IAM roles that you want to remove from the cluster.
5. From **Manage IAM roles**, choose **Remove IAM roles**.

Associating IAM roles with your cluster

You can associate one or more IAM roles with your cluster.

To associate IAM roles with your cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**. The clusters for your account in the current AWS Region are listed. A subset of properties of each cluster is displayed in columns in the list.
3. Choose the cluster that you want to associate IAM roles with.
4. Under **Cluster permissions**, choose one or more IAM roles that you want to associate with the cluster.
5. From **Manage IAM roles**, choose **Associate IAM roles**.
6. Choose one ore more IAM roles to associate with your cluster.
7. Choose **Associate IAM roles**.

Setting an IAM role as the default

You can set an IAM role as the default for your cluster.

To make an IAM role the default for your cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.

2. On the navigation menu, choose **Clusters**. The clusters for your account in the current AWS Region are listed. A subset of properties of each cluster is displayed in columns in the list.
3. Choose the cluster that you want to set a default IAM role for.
4. Under **Cluster permissions**, from **Associated IAM roles**, choose an IAM role that you want make as default for the cluster.
5. Under **Set default**, choose **Make default**.
6. When prompted, choose **Set default** to confirm making the specified IAM role as the default.

Making an IAM role no longer default for your cluster

You can make an IAM role no longer the default for your cluster.

To clear an IAM role as the default for your cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**. The clusters for your account in the current AWS Region are listed. A subset of properties of each cluster is displayed in columns in the list.
3. Choose the cluster that you want to associate IAM roles with.
4. Under **Cluster permissions**, from **Associated IAM roles**, choose the default IAM role.
5. Under **Set default**, choose **Clear default**.
6. When prompted, choose **Clear default** to confirm clearing the specified IAM role as the default.

Managing IAM roles created on the cluster using the AWS CLI

You can manage IAM roles created on the cluster using the AWS CLI.

To create an Amazon Redshift cluster with an IAM role set as default

To create an Amazon Redshift cluster with an IAM role set it as the default for the cluster, use the `aws redshift create-cluster` AWS CLI command.

The following AWS CLI command creates an Amazon Redshift cluster and the IAM role named myrole1. The AWS CLI command also sets myrole1 as the default for the cluster.

```
aws redshift create-cluster \
```

```
    --node-type dc2.large \
    --number-of-nodes 2 \
    --master-username adminuser \
    --master-user-password TopSecret1 \
    --cluster-identifier mycluster \
    --iam-roles 'arn:aws:iam::012345678910:role/myrole1'
 'arn:aws:iam::012345678910:role/myrole2' \
    --default-iam-role-arn 'arn:aws:iam::012345678910:role/myrole1'
```

The following snippet is an example of the response.

```
{
    "Cluster": {
        "ClusterIdentifier": "mycluster",
        "NodeType": "dc2.large",
        "MasterUsername": "adminuser",
        "DefaultIamRoleArn": "arn:aws:iam::012345678910:role/myrole1",
        "IamRoles": [
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole1",
                "ApplyStatus": "adding"
            },
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole2",
                "ApplyStatus": "adding"
            }
        ]
        ...
    }
}
```

To add one or more IAM roles to an Amazon Redshift cluster

To add one or more IAM roles associated to the cluster, use the `aws redshift modify-cluster-iam-roles` AWS CLI command.

The following AWS CLI command adds `myrole3` and `myrole4` to the cluster.

```
aws redshift modify-cluster-iam-roles \
    --cluster-identifier mycluster \
    --add-iam-roles 'arn:aws:iam::012345678910:role/myrole3'
 'arn:aws:iam::012345678910:role/myrole4'
```

The following snippet is an example of the response.

```
{
    "Cluster": {
        "ClusterIdentifier": "mycluster",
        "NodeType": "dc2.large",
        "MasterUsername": "adminuser",
        "DefaultIamRoleArn": "arn:aws:iam::012345678910:role/myrole1",
        "IamRoles": [
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole1",
                "ApplyStatus": "in-sync"
            },
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole2",
                "ApplyStatus": "in-sync"
            },
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole3",
                "ApplyStatus": "adding"
            },
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole4",
                "ApplyStatus": "adding"
            }
        ],
        ...
    }
}
```

To remove one or more IAM roles from an Amazon Redshift cluster

To remove one or more IAM roles associated to the cluster, use the `aws redshift modify-cluster-iam-roles` AWS CLI command.

The following AWS CLI command removes `myrole3` and `myrole4` from the cluster.

```
aws redshift modify-cluster-iam-roles \
    --cluster-identifier mycluster \
    --remove-iam-roles 'arn:aws:iam::012345678910:role/myrole3'
 'arn:aws:iam::012345678910:role/myrole4'
```

The following snippet is an example of the response.

```
{
    "Cluster": {
        "ClusterIdentifier": "mycluster",
        "NodeType": "dc2.large",
        "MasterUsername": "adminuser",
        "DefaultIamRoleArn": "arn:aws:iam::012345678910:role/myrole1",
        "IamRoles": [
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole1",
                "ApplyStatus": "in-sync"
            },
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole2",
                "ApplyStatus": "in-sync"
            },
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole3",
                "ApplyStatus": "removing"
            },
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole4",
                "ApplyStatus": "removing"
            }
        ],
        ...
    }
}
```

To set an associated IAM role as the default for the cluster

To set an associated IAM role as the default for the cluster, use the `aws redshift modify-cluster-iam-roles` AWS CLI command.

The following AWS CLI command sets `myrole2` as the default for the cluster.

```
aws redshift modify-cluster-iam-roles \
    --cluster-identifier mycluster \
    --default-iam-role-arn 'arn:aws:iam::012345678910:role/myrole2'
```

The following snippet is an example of the response.

```
{
```

```
    "Cluster": {
        "ClusterIdentifier": "mycluster",
        "NodeType": "dc2.large",
        "MasterUsername": "adminuser",
        "DefaultIamRoleArn": "arn:aws:iam::012345678910:role/myrole2",
        "IamRoles": [
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole1",
                "ApplyStatus": "in-sync"
            },
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole2",
                "ApplyStatus": "in-sync"
            }
        ],
        ...
    }
}
```

To set an unassociated IAM role as the default for the cluster

To set an unassociated IAM role as the default for the cluster, use the `aws redshift modify-cluster-iam-roles` AWS CLI command.

The following AWS CLI command adds `myrole2` to the Amazon Redshift cluster and sets it as the default for the cluster.

```
aws redshift modify-cluster-iam-roles \
    --cluster-identifier mycluster \
    --add-iam-roles 'arn:aws:iam::012345678910:role/myrole3' \
    --default-iam-role-arn 'arn:aws:iam::012345678910:role/myrole3'
```

The following snippet is an example of the response.

```
{
    "Cluster": {
        "ClusterIdentifier": "mycluster",
        "NodeType": "dc2.large",
        "MasterUsername": "adminuser",
        "DefaultIamRoleArn": "arn:aws:iam::012345678910:role/myrole3",
        "IamRoles": [
            {
```

```
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole1",
                "ApplyStatus": "in-sync"
            },
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole2",
                "ApplyStatus": "in-sync"
            },
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole3",
                "ApplyStatus": "adding"
            }
        ],
        ...
    }
}
```

To restore a cluster from a snapshot and set an IAM role as the default for it

When you restore your cluster from a snapshot, you can either associate an existing IAM role or create a new one and set it as the default for the cluster.

To restore an Amazon Redshift cluster from a snapshot and set an IAM role as the cluster default, use the `aws redshift restore-from-cluster-snapshot` AWS CLI command.

The following AWS CLI command restores the cluster from a snapshot and sets `myrole2` as the default for the cluster.

```
aws redshift restore-from-cluster-snapshot \
    --cluster-identifier mycluster-clone \
    --snapshot-identifier my-snapshot-id
    --iam-roles 'arn:aws:iam::012345678910:role/myrole1'
 'arn:aws:iam::012345678910:role/myrole2' \
    --default-iam-role-arn 'arn:aws:iam::012345678910:role/myrole1'
```

The following snippet is an example of the response.

```
{
    "Cluster": {
        "ClusterIdentifier": "mycluster-clone",
        "NodeType": "dc2.large",
        "MasterUsername": "adminuser",
        "DefaultIamRoleArn": "arn:aws:iam::012345678910:role/myrole1",
```

```
        "IamRoles": [
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole1",
                "ApplyStatus": "adding"
            },
            {
                "IamRoleArn": "arn:aws:iam::012345678910:role/myrole2",
                "ApplyStatus": "adding"
            }
        ],
        ...
    }
}
```

Using a federated identity to manage Amazon Redshift access to local resources and Amazon Redshift Spectrum external tables

Using identity federation in AWS with credentials provided from `GetDatabaseCredentials` can simplify authorization and access to local data and to external data. In this tutorial, we show you how to provide access to resources with AWS identity federation, instead of using a specific IAM role.

Currently, to give users access to external data that resides in Amazon S3, you create an IAM role with permissions defined in a permissions policy. Then, users with the role attached can access the external data. This works, but if you want to provide granular rules, such as making specific columns unavailable for a particular user, you may have to do additional configuration on the external schema.

Identity federation, with credentials provided from `GetDatabaseCredentials`, can provide access to AWS Glue and Redshift Spectrum resources with granular IAM rules that are easier to specify and change. This makes it easier to apply access that conforms to your business rules.

The benefits of using federated credentials are the following:

- You don't have to manage cluster-attached IAM roles for Redshift Spectrum.
- Cluster administrators can create an external schema that's accessible by consumers with different IAM contexts. This is useful, for example, to perform column filtering on a table, where different consumers query the same external schema and get varying fields in returned records.
- You can query Amazon Redshift using a user with IAM permissions, rather than only with a role.

Preparing an identity to log in with federated identity

Before logging in with federated identity, you must perform several preliminary steps. These instructions assume you have an existing Redshift Spectrum external schema that references a data file stored in an Amazon S3 bucket, and the bucket is in the same account as your Amazon Redshift cluster or Amazon Redshift Serverless data warehouse.

1. Create an IAM identity. This can be a user or an IAM role. Use any name supported by IAM.
2. Attach permissions policies to the identity. Specify either of the following:

   - `redshift:GetClusterCredentialsWithIAM` (for an Amazon Redshift provisioned cluster)
   - `redshift-serverless:GetCredentials` (for Amazon Redshift Serverless)

   You can add permissions with the policy editor, using the IAM console.

   The IAM identity also needs permissions to access external data. Grant access to Amazon S3 by adding the following AWS managed policies directly:

   - `AmazonS3ReadOnlyAccess`
   - `AWSGlueConsoleFullAccess`

   The last managed policy is required if you're using AWS Glue to prepare your external data. For more information about the steps for granting access to Amazon Redshift Spectrum, see Create an IAM role for Amazon Redshift, which is part of the getting-started guide for Amazon Redshift and Redshift Spectrum. It shows the steps for adding IAM policies to access Redshift Spectrum.
3. Set up your SQL client to connect to Amazon Redshift. Use the Amazon Redshift JDBC driver, and add your user's credentials to the tool's credential properties. A client like SQL Workbench/J works well for this. Set the following client-connection extended properties:

   - *AccessKeyID* – Your access key identifier.
   - *SecretAccessKey* – Your secret access key. (Note the security risk of transmitting the secret key if you don't use encryption.)
   - *SessionToken* – A set of temporary credentials for an IAM role.

- *groupFederation* – Set to `true` if you're configuring federated identity for a provisioned cluster. Don't set this parameter if you are using Amazon Redshift Serverless.
- *LogLevel* – Integer log-level value. This is optional.

4. Set the URL to the JDBC endpoint found in the Amazon Redshift or Amazon Redshift Serverless console. Replace your URL schema with *jdbc:redshift:iam:* and use this formatting:

   - Format for an Amazon Redshift provisioned cluster: `jdbc:redshift:iam://<cluster_id>.<unique_suffix>.<region>.redshift.amazonaws.com:<port>/<database_name>`

     Example: `jdbc:redshift:iam://test1.12345abcdefg.us-east-1.redshift.amazonaws.com:5439/dev`
   - Format for Amazon Redshift Serverless: `jdbc:redshift:iam://<workgroup-name>.<account-number>.<aws-region>.redshift-serverless.amazonaws.com:5439:<port>/<database_name>`

     Example: `jdbc:redshift:iam://default.123456789012.us-east-1.redshift-serverless.amazonaws.com:5439/dev`

   After you connect to the database for the first time, using an IAM identity, Amazon Redshift automatically creates an Amazon Redshift identity with the same name, prefixed with `IAM:` for a user or `IAMR:` for an IAM role. The remaining steps in this topic show examples for a user.

   If a Redshift user isn't automatically created, you can create one by running a `CREATE USER` statement, using an admin account, specifying the user name in the format `IAM:<user name>`.

5. As your Amazon Redshift cluster administrator, grant the Redshift user the required permissions to access the external schema.

   ```
   GRANT ALL ON SCHEMA my_schema to "IAM:my_user";
   ```

   To grant the ability to your Redshift user to create tables in the external schema, they must be a schema owner. For example:

   ```
   ALTER SCHEMA my_schema owner to "IAM:my_user";
   ```

6. To verify the configuration, run a query as the user, using the SQL client, after permissions are granted. This query sample retrieves data from an external table.

```
SELECT * FROM my_schema.my_table;
```

Getting started with identity and authorization propagation to Redshift Spectrum

To pass a federated identity to query external tables, you set `SESSION` as the value for the `IAM_ROLE` query parameter of `CREATE EXTERNAL SCHEMA`. The following steps show how to set up and leverage `SESSION` to authorize queries on the external schema.

1. Create local tables and external tables. External tables catalogued with AWS Glue work for this.
2. Connect to Amazon Redshift with your IAM identity. As mentioned in the previous section, when the identity connects to Amazon Redshift, a Redshift database user is created. The user is created if they didn't previously exist. If the user is new, the administrator must grant them permissions to perform tasks in Amazon Redshift, like querying and creating tables.
3. Connect to Redshift with your admin account. Run the command to create an external schema, using the `SESSION` value.

```
create external schema spectrum_schema from data catalog
database '<my_external_database>'
region '<my_region>'
iam_role 'SESSION'
catalog_id '<my_catalog_id>';
```

Note that `catalog_id` is set in this case. This is a new setting added with the feature, because `SESSION` replaces a specific role.

In this example, values in the query mimic how real values appear.

```
create external schema spectrum_schema from data catalog
database 'spectrum_db'
region 'us-east-1'
iam_role 'SESSION'
catalog_id '123456789012'
```

The `catalog_id` value in this case is your AWS account ID.

4. Run queries to access your external data, using the IAM identity you connected with in step 2. For example:

   ```
   select * from spectrum_schema.table1;
   ```

   In this case, `table1` can be, for example, JSON-formatted data in a file, in an Amazon S3 bucket.

5. If you already have an external schema that uses a cluster-attached IAM role, pointing to your external database or schema, you can either replace the existing schema and use a federated identity as detailed in these steps, or create a new one.

`SESSION` indicates that federated identity credentials are used to query the external schema. When you use the `SESSION` query parameter, make sure you set the `catalog_id`. It's required because it points to the data catalog used for the schema. Previously, `catalog_id` was retrieved from the value assigned to `iam_role`. When you set up identity and authorization propagation this way, for instance, to Redshift Spectrum, by using federated credentials to query an external schema, authorization by means of an IAM role isn't required.

Usage notes

A common connection error is the following: *IAM error retrieving temp credentials: Unable to unmarshall exception response with the unmarshallers provided*. This error is a result of having a legacy JDBC driver. The minimum driver version required for federated identity is 2.1.0.9. You can get the JDBC driver from Download the Amazon Redshift JDBC driver, version 2.x.

Additional resources

These links provide additional information for managing access to external data.

- You can still access Redshift Spectrum data using an IAM role. For more information, see Authorizing Amazon Redshift to access AWS services on your behalf.
- When you manage access to external tables with AWS Lake Formation, you can query them using Redshift Spectrum with federated IAM identities. You no longer have to manage cluster-attached IAM roles for Redshift Spectrum to query data registered with AWS Lake Formation. For more information, see Using AWS Lake Formation with Amazon Redshift Spectrum.

Managing Amazon Redshift admin passwords using AWS Secrets Manager

Amazon Redshift can integrate with AWS Secrets Manager to generate and manage your admin credentials inside an encrypted secret. With AWS Secrets Manager, you can replace your admin passwords with an API call to programmatically retrieve the secret when it's needed. Using secrets instead of hard-coded credentials reduces the risk of those credentials being exposed or compromised. For more information about AWS Secrets Manager, see the *AWS Secrets Manager User Guide*.

You can specify that Amazon Redshift manages your admin password using AWS Secrets Manager when you perform one of the following operations:

- Create a provisioned cluster or serverless namespace
- Edit, update, or modify the admin credentials of a provisioned cluster or serverless namespace
- Restore a cluster or serverless namespace from a snapshot

When you specify that Amazon Redshift manages the admin password in AWS Secrets Manager, Amazon Redshift generates the password and stores it in Secrets Manager. You can access the secret directly in AWS Secrets Manager to retrieve the credentials for the admin user. Optionally, you can specify a customer managed key to encrypt the secret if you need to access the secret from another AWS account. You can also use the KMS key that AWS Secrets Manager provides.

Amazon Redshift manages the settings for the secret and rotates the secret every 30 days by default. You can manually rotate the secret at any time. If you delete a provisioned cluster or serverless namespace that manages a secret in AWS Secrets Manager, the secret and its associated metadata are also deleted.

To connect to a cluster or serverless namespace with secret-managed credentials, you can retrieve the secret from AWS Secrets Manager using the Secrets Manager console or the `GetSecretValue` Secrets Manager API call. For more information, see Retrieve secrets from AWS Secrets Manager and Connect to a SQL database with credentials in an AWS Secrets Manager secret in the *AWS Secrets Manager User Guide*.

Permissions required for AWS Secrets Manager integration

Users must have the required permissions to perform operations related to AWS Secrets Manager integration. Create IAM policies that grant permissions to perform specific API operations on the

specified resources they need. Then attach those policies to the IAM permission sets or roles that require those permissions. For more information, see Identity and access management in Amazon Redshift.

The user who specifies that Amazon Redshift manages the admin password in AWS Secrets Manager must have permissions to perform the following operations:

- `secretsmanager:CreateSecret`
- `secretsmanager:RotateSecret`
- `secretsmanager:DescribeSecret`
- `secretsmanager:UpdateSecret`
- `secretsmanager:DeleteSecret`
- `secretsmanager:GetRandomPassword`
- `secretsmanager:TagResource`

If the user wants to pass a KMS key in the `MasterPasswordSecretKmsKeyId` parameter for provisioned clusters, or the `AdminPasswordSecretKmsKeyId` parameter for serverless namespaces, they require the following permissions in addition to the permissions listed above.

- `kms:Decrypt`
- `kms:GenerateDataKey`
- `kms:CreateGrant`
- `kms:RetireGrant`

Admin password secret rotation

By default, Amazon Redshift automatically rotates your secret every 30 days to ensure your credentials don't stay the same for prolonged periods. When Amazon Redshift rotates an admin password secret, AWS Secrets Manager updates the existing secret to contain a new admin password. Amazon Redshift changes the admin password for the cluster to match the password in the updated secret.

You can rotate a secret immediately instead of waiting for a scheduled rotation by using AWS Secrets Manager. For more information on rotating secrets, see Rotate AWS Secrets Manager secrets in the *AWS Secrets Manager User Guide*.

Considerations using AWS Secrets Manager with Amazon Redshift

When using AWS Secrets Manager to manage your provisioned cluster or serverless namespace's admin credentials, consider the following:

- When you pause a cluster whose admin credentials are managed by AWS Secrets Manager, your cluster's secret won't be deleted and you'll continue to be billed for the secret. Secrets are only deleted when you delete the cluster.
- If your cluster is paused when Amazon Redshift attempts to rotate its attached secret, the rotation will fail. In this case, Amazon Redshift stops auto-rotation and won't try to rotate it again, even after you resume the cluster. You must restart the auto-rotation schedule using the `secretsmanager:RotateSecret` API call to continue having AWS Secrets Manager automatically rotate your secret.
- If your serverless namespace doesn't have a workgroup associated when Amazon Redshift attempts to rotate its attached secret, the rotation will fail and won't try to rotate it again, even after you attach a workgroup. You must restart the auto-rotation schedule using the `secretsmanager:RotateSecret` API call to continue having AWS Secrets Manager automatically rotate your secret.

Retrieving the Amazon Resource Name (ARN) of the secret in Amazon Redshift

You can view the Amazon Resource Name (ARN) for any secrets being managed by AWS Secrets Manager using the Amazon Redshift console. Once you have the secret's ARN, you can view details about your secret and the encrypted data in your secret using AWS Secrets Manager. For more information on retrieving secrets using the ARN, see Retrieve secrets in the *AWS Secrets Manager User Guide*.

Viewing the details about a secret for an Amazon Redshift provisioned cluster

View the Amazon Resource Name (ARN) for your cluster's secret using the Amazon Redshift console with the following procedure:

1. Sign in to the AWS Management Console and open the Amazon Redshift console.
2. In the **Cluster overview** pane, choose the cluster whose secret you want to view.
3. Choose the **Properties** tab.

4. View the secret's ARN under **Admin credentials ARN**. This ARN is the identifier for the secret, which you can use in AWS Secrets Manager to view the secret's details.

Viewing the details about a secret for an Amazon Redshift Serverless namespace

View the Amazon Resource Name (ARN) for your serverless namespace's secret using the Amazon Redshift console with the following procedure:

1. Sign in to the AWS Management Console and open the Amazon Redshift console.
2. From the **Provisioned clusters** dashboard, choose **Go to Serverless** in the upper right of the page.
3. From the **Serverless dashboard**, scroll to the **Namespaces / Workgroups** pane and choose the namespace whose secret you want to view.
4. In the **General information** pane, view the secret's ARN under **Admin credentials ARN**. This ARN is the identifier for the secret, which you can use in AWS Secrets Manager to view the secret's details.

Creating a secret for database connection credentials

You can create a Secrets Manager secret to store credentials used to connect to an Amazon Redshift provisioned cluster or Redshift Serverless namespace and workgroup. You can also use this secret when scheduling a query in Amazon Redshift query editor v2.

To create a secret for a database in an Amazon Redshift provisioned cluster using the Secrets Manager console

1. Open the Secrets Manager console (https://console.aws.amazon.com/secretsmanager/).
2. Navigate to the list of **Secrets** and choose **Store a new secret**.
3. Choose **Credentials for Amazon Redshift data warehouse**. Enter your information in the steps to create a secret as follows:

   - In **Credentials** for **User name**, enter the name of the administrative user of the data warehouse.
   - In **Credentials** for **Password**, enter the password for the **User name**.
   - For **Encryption key**, choose your encryption key.
   - For **Data warehouse**, choose the Amazon Redshift provisioned cluster that contains your data.
   - For **Secret name**, enter a name for the secret.

- For **Description**, enter a description of the secret.
- For **Tags**, enter a **Tag key** with the word **`Redshift`**. This tag key is needed to list secrets when you attempt to connect to your data warehouse using Amazon Redshift query editor v2. The secret must have a tag key that starts with the string **`Redshift`** for the secret to be listed under AWS Secrets Manager on the management console.

4. Continue entering information about your secret through several steps until you **Store** your changes on the **Review** step.

   The specific values of your credentials, engine, host, port, and cluster identifier are stored in the secret. Also, the secret is tagged with the tag key `Redshift`.

To create a secret for a database in a Redshift Serverless namespace using the Redshift Serverless console

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. Choose **Redshift serverless** and navigate to **Namespace configuration**.
3. Choose a namespace for which to create secret credentials.
4. Open **Actions**, **Edit admin credentials**.
5. For **Admin password**, choose **Manage admin credentials in AWS Secrets Manager**.
6. Choose **Save changes** to save your changes.

Confirm that a message appears that the password successfully changed. You can also view the secret in the Secrets Manager console. You can use this secret to connect to a database in a workgroup in the Redshift Serverless console and Amazon Redshift query editor v2, using the AWS Secrets Manager connection method. The secret must have a tag key that starts with the string "Redshift" for the secret to be listed on the query editor v2 web application. The secret must have a tag key that starts with the string **`Redshift`** for the secret to be listed under AWS Secrets Manager on the management console.

To create a secret for a database in a Redshift Serverless namespace using the Secrets Manager console

1. Open the Secrets Manager console (https://console.aws.amazon.com/secretsmanager/).
2. Navigate to the list of **Secrets** and choose **Store a new secret**.

3. Choose **Credentials for Amazon Redshift data warehouse**. Enter your information in the steps to create a secret as follows:

   - In **Credentials** for **User name**, enter the name of the administrative user of the data warehouse.
   - In **Credentials** for **Password**, enter the password for the **User name**.
   - For **Encryption key**, choose your encryption key.
   - For **Data warehouse**, choose the Redshift Serverless namespace that contains your data.
   - For **Secret name**, enter a name for the secret.
   - For **Description**, enter a description of the secret.
   - For **Tags**, enter a **Tag key** with the word **`Redshift`**. This tag key is needed to list secrets when you attempt to connect to your data warehouse using Amazon Redshift query editor v2. The secret must have a tag key that starts with the string **`Redshift`** for the secret to be listed under AWS Secrets Manager on the management console.
4. Continue entering information about your secret through several steps until you **Store** your changes on the **Review** step.

   The specific values of your credentials, database name, host, port, namespace, and engine are stored in the secret. Also, the secret is tagged with the tag key `Redshift`.

To create a secret for a database in a Redshift Serverless namespace using the AWS CLI

You can use the AWS CLI to create a secret. One method is to use AWS CloudShell to run the Secrets Manager AWS CLI command as follows. You must have the proper permissions to run the AWS CLI commands shown in the following procedure.

1. On the AWS console, open the AWS CloudShell command prompt. For more information about AWS CloudShell, see What is AWS CloudShell in the *AWS CloudShell User Guide*.
2. For example, for the secret `MyTestSecret` enter an Secrets Manager command to store the secret that is used to connect to a database or schedule an Amazon Redshift query editor v2 query. Replace the following values in the command with values for your environment:

   - *`admin`* is the administrator user name for the data warehouse.
   - *`passw0rd`* is the password of the administrator.
   - *`dev`* is the initial database name in the data warehouse.
   - *`region`* is the AWS Region that contains the data warehouse. For example `us-east-1`.

- *123456789012* is the AWS account.
- *namespace-id* is the namespace identifier similar to c3928f0e-c889-4d2b-97a5-5738324d5d3e. You can find this identifier on the Amazon Redshift console details page for the serverless namespace.

```
aws secretsmanager create-secret \
--name MyTestSecret \
--description "My test secret created with the CLI." \
--secret-string "{\"username\":\"admin\",\"password\":\"passw0rd\",\"dbname\":
\"dev\",\"engine\":\"redshift\"}" \
--tags "[{\"Key\":\"redshift-serverless:namespaceArn\",\"Value\":
\"arn:aws:redshift-serverless:region:123456789012:namespace/namespace-id\"}]"
```

Logging and monitoring in Amazon Redshift

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon Redshift and your AWS solutions. You can collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. AWS provides several tools for monitoring your Amazon Redshift resources and responding to potential incidents:

Amazon CloudWatch Alarms

Using Amazon CloudWatch alarms, you watch a single metric over a time period that you specify. If the metric exceeds a given threshold, a notification is sent to an Amazon SNS topic or AWS Auto Scaling policy. CloudWatch alarms do not invoke actions because they are in a particular state. Rather the state must have changed and been maintained for a specified number of periods. For more information, see Creating an alarm. For a list of metrics, see Performance data in Amazon Redshift.

AWS CloudTrail Logs

CloudTrail provides a record of API operations taken by a user, an IAM role, or an AWS service in Amazon Redshift. 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. For more information, see Logging with CloudTrail.

Database audit logging

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, a process called *database auditing*. The logs can be stored in:

- *Amazon S3 buckets* - This provides access with data-security features for users who are responsible for monitoring activities in the database.
- *Amazon CloudWatch* - You can view audit-logging data using the features built into CloudWatch, such as visualization features and setting actions.

> **Note**
>
> SYS_CONNECTION_LOG collects connection log data for Amazon Redshift Serverless. Note that when you collect audit logging data for Amazon Redshift Serverless, it can't be sent to log files, only to CloudWatch.

Topics

- Amazon Redshift logs
- Audit logs and Amazon CloudWatch
- Enabling audit logging
- Secure logging

Amazon Redshift logs

Amazon Redshift logs information in the following log files:

- *Connection log* – Logs authentication attempts, connections, and disconnections.
- *User log* – Logs information about changes to database user definitions.
- *User activity log* – Logs each query before it's 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 users connecting to the database and related connection information. This information might be 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 a simpler 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.

Note

Log files are not as current as the system log tables which are STL_USERLOG and STL_CONNECTION_LOG. Records that are older than, but not including, the latest record are copied to log files.

Note

For Amazon Redshift Serverless, SYS_CONNECTION_LOG collects connection-log data. When you collect audit logging data for Amazon Redshift Serverless, it can't be sent to log files, only to CloudWatch.

Connection log

Logs authentication attempts, and connections and disconnections. The following table describes the information in the connection log. For more information about these fields, see STL_CONNECTION_LOG in the *Amazon Redshift Database Developer Guide*. For more information about the collected connection log data for Amazon Redshift Serverless, see SYS_CONNECTION_LOG.

Column name	Description
event	Connection or authentication event.

Column name	Description
recordtime	Time the event occurred.
remotehost	Name or IP address of remote host.
remoteport	Port number for remote host.
pid	Process ID associated with the statement.
dbname	Database name.
username	User name.
authmethod	Authentication method.
duration	Duration of connection in microseconds.
sslversion	Secure Sockets Layer (SSL) version.
sslcipher	SSL cipher.
mtu	Maximum transmission unit (MTU).
sslcompression	SSL compression type.
sslexpansion	SSL expansion type.
iamauthguid	The AWS Identity and Access Management (IAM) authentication ID for the AWS CloudTrail request. This is the identifier for the GetClusterCredentials API call to create the credentials being used for a given connection.
application_name	The initial or updated name of the application for a session.
os_version	The version of the operating system that is on the client machine that connects to your Amazon Redshift cluster.
driver_version	The version of ODBC or JDBC driver that connects to your Amazon Redshift cluster from your third-party SQL client tools.

Column name	Description
plugin_name	The name of the plugin used to connect to your Amazon Redshift cluster.
protocol_ version	The internal protocol version that the Amazon Redshift driver uses when establishing its connection with the server.
sessionid	The globally unique identifier for the current session.
compression	The compression algorithm in use for the connection.

User log

Records details for the following changes to a database user:

- Create user
- Drop user
- Alter user (rename)
- Alter user (alter properties)

Column name	Description
userid	ID of user affected by the change.
username	User name of the user affected by the change.
oldusername	For a rename action, the original user name. For any other action, this field is empty.
action	Action that occurred. Valid values: • Alter • Create • Drop • Rename
usecreatedb	If true (1), indicates that the user has create database permissions.

Column name	Description
usesuper	If true (1), indicates that the user is a superuser.
usecatupd	If true (1), indicates that the user can update system catalogs.
valuntil	Password expiration date.
pid	Process ID.
xid	Transaction ID.
recordtime	Time in UTC that the query started.

Query the SYS_USERLOG system view to find additional information about changes to users. This view includes log data from Amazon Redshift Serverless.

User activity log

Logs each query before it is run on the database.

Column name	Description
recordtime	Time the event occurred.
db	Database name.
user	User name.
pid	Process ID associated with the statement.
userid	User ID.
xid	Transaction ID.
query	A prefix of LOG: followed by the text of the query, including newlines.

Audit logs and Amazon CloudWatch

Audit logging is not turned on by default in Amazon Redshift. When you turn on logging on your cluster, Amazon Redshift exports logs to Amazon CloudWatch, or creates and uploads logs to Amazon S3, that capture data from the time audit logging is enabled to the present time. Each logging update is a continuation of the previous logs.

Audit logging to CloudWatch or to Amazon S3 is an optional process. Logging to system tables is not optional and happens automatically. For more information about logging to system tables, see System Tables Reference in the Amazon Redshift Database Developer Guide.

The connection log, user log, and user activity log are enabled together by using the AWS Management Console, the Amazon Redshift API Reference, or the AWS Command Line Interface (AWS CLI). For the user activity log, you must also enable the `enable_user_activity_logging` database parameter. If you enable only the audit logging feature, but not the associated parameter, the database audit logs log information for only the connection log and user log, but not for the user activity log. The `enable_user_activity_logging` parameter is not enabled (`false`) by default. You can set it to `true` to enable the user activity log. For more information, see Amazon Redshift parameter groups.

When you enable logging to CloudWatch, Amazon Redshift exports cluster connection, user, and user-activity log data to an Amazon CloudWatch Logs log group. The log data doesn't change, in terms of schema. CloudWatch is built for monitoring applications, and you can use it to perform real-time analysis or set it to take actions. You can also use Amazon CloudWatch Logs to store your log records in durable storage.

Using CloudWatch to view logs is a recommended alternative to storing log files in Amazon S3. It doesn't require much configuration, and it may suit your monitoring requirements, especially if you use it already to monitor other services and applications.

Log groups and log events in Amazon CloudWatch

After selecting which Amazon Redshift logs to export, you can monitor log events in Amazon CloudWatch Logs. A new log group is automatically created for Amazon Redshift Serverless, under the following prefix, in which `log_type` represents the log type.

```
/aws/redshift/cluster/<cluster_name>/<log_type>
```

For example, if you choose to export the connection log, log data is stored in the following log group.

```
/aws/redshift/cluster/cluster1/connectionlog
```

Log events are exported to a log group using the log stream. To search for information within log events for your serverless endpoint, use the Amazon CloudWatch Logs console, the AWS CLI, or the Amazon CloudWatch Logs API. For information about searching and filtering log data, see Creating metrics from log events using filters.

In CloudWatch, you can search your log data with a query syntax that provides for granularity and flexibility. For more information, see CloudWatch Logs Insights query syntax.

Migrating to Amazon CloudWatch audit logging

In any case where you are sending logs to Amazon S3 and you change the configuration, for example to send logs to CloudWatch, logs that remain in Amazon S3 are unaffected. You can still query the log data in the Amazon S3 buckets where it resides.

Log files in Amazon S3

The number and size of Amazon Redshift log files in Amazon S3 depends heavily on the activity in your cluster. If you have an active cluster that is generating a large number of logs, Amazon Redshift might generate the log files more frequently. You might have a series of log files for the same type of activity, such as having multiple connection logs within the same hour.

When Amazon Redshift uses Amazon S3 to store logs, you incur charges for the storage that you use in Amazon S3. Before you configure logging to Amazon S3, plan for how long you need to store the log files. As part of this, determine when the log files can either be deleted or archived, based on your auditing needs. The plan that you create depends heavily on the type of data that you store, such as data subject to compliance or regulatory requirements. For more information about Amazon S3 pricing, go to Amazon Simple Storage Service (S3) Pricing.

Limitations when you enable logging to Amazon S3

Audit logging has the following constraints:

- You can use only Amazon S3-managed keys (SSE-S3) encryption (AES-256).
- The Amazon S3 buckets must have the S3 Object Lock feature turned off.

Bucket permissions for Amazon Redshift audit logging

When you turn on logging to Amazon S3, 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. Also, the user or IAM role that turns on logging must have `s3:PutObject` permission to the Amazon S3 bucket. 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 don't match, you receive an error.

If, when you enable audit logging, you select the option to create a new bucket, correct permissions are applied to it. However, if you create your own bucket in Amazon S3, or use an existing bucket, make sure to add a bucket policy that includes the bucket name. Logs are delivered using service-principal credentials. For most AWS Regions, you add the Redshift service-principal name, *`redshift.amazonaws.com`*.

The bucket policy uses the following format. *`ServiceName`* and *`BucketName`* are placeholders for your own values. Also specify the associated actions and resources in the bucket policy.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Put bucket policy needed for audit logging",
            "Effect": "Allow",
            "Principal": {
                "Service": "ServiceName"
            },
            "Action": [
                "s3:PutObject",
                "s3:GetBucketAcl"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::BucketName",
                "arn:aws:s3:::BucketName/*"
            ]
        }
    ]
}
```

The following example is a bucket policy for the US East (N. Virginia) Region and a bucket named `AuditLogs`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Put bucket policy needed for audit logging",
            "Effect": "Allow",
            "Principal": {
                "Service": "redshift.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:GetBucketAcl"
            ],
            "Resource": [
                "arn:aws:s3:::AuditLogs",
                "arn:aws:s3:::AuditLogs/*"
            ]
        }
    ]
}
```

Regions that aren't enabled by default, also known as "opt-in" Regions, require a Region-specific service principal name. For these, the service-principal name includes the region, in the format `redshift.`*`region`*`.amazonaws.com`. For example, *`redshift.ap-east-1.amazonaws.com`* for the Asia Pacific (Hong Kong) Region. For a list of the Regions that aren't enabled by default, see Managing AWS Regions in the *AWS General Reference*.

> **Note**
>
> The Region-specific service-principal name corresponds to the Region where the cluster is located.

Best practices for log files

When Redshift uploads log files to Amazon S3, large files can be uploaded in parts. If a multipart upload isn't successful, it's possible for parts of a file to remain in the Amazon S3 bucket. This can result in additional storage costs, so it's important to understand what occurs when a multipart upload fails. For a detailed explanation about multipart upload for audit logs, see Uploading and copying objects using multipart upload and Aborting a multipart upload.

For more information about creating S3 buckets and adding bucket policies, see Creating a general purpose bucket and Bucket policies for Amazon S3 in the *Amazon Simple Storage Service User Guide*.

Bucket structure for Amazon Redshift audit logging

By default, Amazon Redshift organizes the log files in the Amazon S3 bucket by using the following bucket and object structure:

`AWSLogs/`*`AccountID`*`/`*`ServiceName`*`/`*`Region`*`/`*`Year`*`/`*`Month`*`/`*`Day`*`/`*`AccountID_ServiceName_Region`*

An example is: `AWSLogs/123456789012/redshift/us-east-1/2013/10/29/123456789012_redshift_us-east-1_mycluster_userlog_2013-10-29T18:01.gz`

If you provide an Amazon S3 key prefix, put the prefix at the start of the key.

For example, if you specify a prefix of myprefix: `myprefix/AWSLogs/123456789012/redshift/us-east-1/2013/10/29/123456789012_redshift_us-east-1_mycluster_userlog_2013-10-29T18:01.gz`

The Amazon S3 key prefix can't exceed 512 characters. It can't contain spaces ( ), double quotation marks (“), single quotation marks (‘), a backslash (\). There is also a number of special characters and control characters that aren't allowed. The hexadecimal codes for these characters are as follows:

- x00 to x20
- x22
- x27
- x5c
- x7f or larger

Audit logging in Amazon S3 considerations

Amazon Redshift audit logging can be interrupted for the following reasons:

- Amazon Redshift does not have permission to upload logs to the Amazon S3 bucket. Verify that the bucket is configured with the correct IAM policy. For more information, see Bucket permissions for Amazon Redshift audit logging.
- The bucket owner changed. When Amazon Redshift uploads logs, it verifies that the bucket owner is the same as when logging was enabled. If the bucket owner has changed, Amazon Redshift cannot upload logs until you configure another bucket to use for audit logging.
- The bucket cannot be found. If the bucket is deleted in Amazon S3, Amazon Redshift cannot upload logs. You either must recreate the bucket or configure Amazon Redshift to upload logs to a different bucket.

API calls with AWS CloudTrail

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. For more information about Amazon Redshift integration with AWS CloudTrail, see Logging with CloudTrail.

You can use CloudTrail independently from or in addition to Amazon Redshift database audit logging.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Enabling audit logging

Configure Amazon Redshift to export audit log data. Logs can be exported to CloudWatch, or as files to Amazon S3 buckets.

Enabling audit logging using the console

Console steps

To enable audit logging for a cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**, then choose the cluster that you want to update.
3. Choose the **Properties** tab. On the **Database configurations** panel, choose **Edit**, then **Edit audit logging**.
4. On the **Edit audit logging** page, choose **Turn on** and select **S3 bucket** or **CloudWatch**. We recommend using CloudWatch because administration is easy and it has helpful features for data visualization.
5. Choose which logs to export.
6. To save your choices, choose **Save changes**.

Secure logging

When Amazon Redshift logs a query that references one or more AWS Glue Data Catalog views, Amazon Redshift automatically masks fields in certain system table and view columns when logging metadata about that query.

Secure log masking applies to all system table and view entries that Amazon Redshift generates while running a query that fits the masking conditions. The following table lists system views and columns that have secure logging applied, masking text with `******` and numbers with `-1`. The number of asterisks used to mask text matches the number of characters in the original text, up to 6 characters. Strings longer than 6 characters still appear as 6 asterisks.

System table	Sensitive columns
SYS_EXTERNAL_QUERY_DETAIL	**Columns:** source_type, total_partitions, qualified_partitions, scanned_files, returned_rows, returned_bytes, file_format, file_location, external_query_text, warning_message.
SYS_EXTERNAL_QUERY_ERROR	**Columns:** file_location, rowid, column_name, original_value, modified_value, trigger, action, action_value, error_code.

System table	Sensitive columns
SYS_QUERY_DETAIL	**Columns:** step_id, step_name, table_id, table_name, input_bytes, input_rows, output_bytes, output_rows, blocks_read, blocks_write, local_read_IO, remote_read_IO, spilled_block_local_disk, spilled_block_remote_disk, step_attribute.
SYS_QUERY_HISTORY	**Columns:** returned_rows, returned_bytes.
STL_AGGR	**Columns:** rows, bytes, tbl, type.
STL_BCAST	**Columns:** rows, bytes, packets.
STL_DDLTEXT	**Columns:** label, text.
STL_DELETE	**Columns:** rows, tbl.
STL_DIST	**Columns:** rows, bytes, packets.
STL_ERROR	**Columns:** file, linenum, context, error.
STL_EXPLAIN	**Columns:** plannode, info.
STL_FILE_SCAN	**Columns:** name, line, bytes.
STL_HASH	**Columns:** rows, bytes, tbl, est_rows.
STL_HASHJOIN	**Columns:** rows, tbl, num_parts, join_type.
STL_INSERT	**Columns:** rows, tbl.
STL_LIMIT	**Columns:** rows.
STL_MERGE	**Columns:** rows.
STL_MERGEJOIN	**Columns:** rows, tbl.
STL_NESTLOOP	**Columns:** rows, tbl.
STL_PARSE	**Columns:** rows.
STL_PLAN_INFO	**Columns:** startupcost, totalcost, rows, bytes.

System table	Sensitive columns
STL_PROJECT	**Columns:** rows, tbl.
STL_QUERY	**Columns:** querytxt.
STL_QUERY_METRICS	**Columns:** max_rows, rows, max_blocks_read, blocks_read, max_blocks_to_disk, blocks_to_disk, max_query_scan_size, query_scan_size.
STL_QUERYTEXT	**Columns:** text.
STL_RETURN	**Columns:** rows, bytes.
STL_S3CLIENT	**Columns:** bucket, key, transfer_size, data_size.
STL_S3CLIENT_ERROR	**Columns:** bucket, key, error, transfer_size.
STL_SAVE	**Columns:** rows, bytes, tbl.
STL_SCAN	**Columns:** rows, bytes, fetches, type, tbl, rows_pre_filter, rows_pre_user_filter, perm_table_name, scanned_mega_value.
STL_SORT	**Columns:** rows, bytes, tbl.
STL_SSHCLIENT_ERROR	**Columns:** ssh_username, endpoint, command, error.
STL_TR_CONFLICT	**Columns:** table_id.
STL_UNDONE	**Columns:** table_id.
STL_UNIQUE	**Columns:** rows, type, bytes.
STL_UTILITYTEXT	**Columns:** label, text.
STL_WINDOW	**Columns:** rows.
STV_BLOCKLIST	**Columns:** col, tbl, num_values, minvalue, maxvalue.
STV_EXEC_STATE	**Columns:** rows, bytes, label.
STV_INFLIGHT	**Columns:** label, text.

System table	Sensitive columns
STV_LOCKS	**Columns:** table_id.
STV_QUERY_METRICS	**Columns:** rows, max_rows, blocks_read, max_blocks_read, max_blocks_to_disk, blocks_to_disk, max_query_scan_size, query_scan_size.
STV_STARTUP_RECOVERY_STATE	**Columns:** table_id, table_name.
STV_TBL_PERM	**Columns:** id, name, rows, sorted_rows, temp, block_count, query_scan_size.
STV_TBL_TRANS	**Columns:** id, rows, size.
SVCS_EXPLAIN	**Columns:** plannode, info.
SVCS_PLAN_INFO	**Columns:** rows, bytes.
SVCS_QUERY_SUMMARY	**Columns:** step, rows, bytes, rate_row, rate_byte, label, rows_pre_filter.
SVCS_S3LIST	**Columns:** bucket, prefix, retrieved_files, max_file_size, avg_file_size.
SVCS_S3LOG	**Columns:** message.
SVCS_S3PARTITION_SUMMARY	**Columns:** total_partitions, qualified_partitions, min_assigned_partitions, max_assigned_partitions, avg_assigned_partitions.
SVCS_S3QUERY_SUMMARY	**Columns:** external_table_name, file_format, s3_scanned_rows, s3_scanned_bytes, s3query_returned_rows, s3query_returned_bytes.
SVL_QUERY_METRICS	**Columns:** step_label, scan_row_count, join_row_count, nested_loop_join_row_count, return_row_count, spectrum_scan_row_count, spectrum_scan_size_mb.
SVL_QUERY_METRICS_SUMMARY	**Columns:** step_label, scan_row_count, join_row_count, nested_loop_join_row_count, return_row_count, spectrum_scan_row_count, spectrum_scan_size_mb.

System table	Sensitive columns
SVL_QUERY_REPORT	**Columns:** rows, bytes, label, rows_pre_filter.
SVL_QUERY_SUMMARY	**Columns:** rows, bytes, rows_pre_filter.
SVL_S3LIST	**Columns:** bucket, prefix, retrieved_files, max_file_size, avg_file_size.
SVL_S3LOG	**Columns:** message.
SVL_S3PARTITION	**Columns:** rows, bytes, label, rows_pre_filter.
SVL_S3PARTITION_SUMMARY	**Columns:** total_partitions, qualified_partitions, min_assigned_partitions, max_assigned_partitions, avg_assigned_partitions.
SVL_S3QUERY	**Columns:** external_table_name, file_format, s3_scanned_rows, s3_scanned_bytes, s3query_returned_rows, s3query_returned_bytes, files.
SVL_S3QUERY_SUMMARY	**Columns:** external_table_name, file_format, s3_scanned_rows, s3_scanned_bytes, s3query_returned_rows, s3query_returned_bytes.
SVL_S3RETRIES	**Columns:** file_size, location, message.
SVL_SPECTRUM_SCAN_ERROR	**Columns:** location, rowid, colname, original_value, modified_value, trigger, action, action_value, error_code.
SVL_STATEMENTTEXT	**Columns:** type, text.
SVL_STORED_PROC_CALL	**Columns:** querytxt.
SVL_STORED_PROC_MESSAGES	**Columns:** message, linenum, querytext.
SVL_UDF_LOG	**Columns:** message, funcname.
SVV_DISKUSAGE	**Columns:** name, col, tbl, blocknum, num_values, minvalue, maxvalue.
SVV_QUERY_STATE	**Columns:** rows, bytes, label.

System table	Sensitive columns
SVV_TABLE_INFO	**Columns:** table_id, table.
SVV_TRANSACTIONS	**Columns:** relation.

For more information on system tables and views, see System tables and views reference in the *Amazon Redshift Database Developer Guide*. For information on Amazon Redshift's ability to dynamically mask query results, see Dynamic data masking in the *Amazon Redshift Database Developer Guide*. For information on creating views in the AWS Glue Data Catalog using Amazon Redshift, see AWS Glue Data Catalog views in the *Amazon Redshift Database Developer Guide*.

Logging with CloudTrail

Amazon Redshift, data sharing, Amazon Redshift Serverless, Amazon Redshift Data API, and query editor v2 are all integrated with AWS CloudTrail. CloudTrail is 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. The calls captured include calls from the Redshift console and code calls to the Redshift operations.

If you create a CloudTrail trail, you can have continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for 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 certain things. These include the request that was made to Redshift, the IP address from which the request was made, who made the request, when it was made, and additional details.

You can use CloudTrail independently from or in addition to Amazon Redshift database audit logging.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Information in CloudTrail

CloudTrail is turned on in your AWS account when you create the account. When activity occurs, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History in the *AWS CloudTrail User Guide*.

For an ongoing record of events in your AWS account, including events for Redshift, create a trail. CloudTrail uses *trails* to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following in the *AWS CloudTrail User Guide*:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

All Amazon Redshift, Amazon Redshift Serverless, Data API, data sharing, and query editor v2 actions are logged by CloudTrail. For example, calls to the `AuthorizeDatashare`, `CreateNamespace`, `ExecuteStatement`, and `CreateConnection` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see CloudTrail userIdentity Element in the *AWS CloudTrail User Guide*.

Log files entries

A *trail* is a configuration that allows for delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An *event* represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

Amazon Redshift Datashare example

The following example shows a CloudTrail log entry that illustrates the `AuthorizeDataShare` operation.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE:janedoe",
        "arn": "arn:aws:sts::111122223333:user/janedoe",
        "accountId": "111122223333",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AKIAIOSFODNN7EXAMPLE:janedoe",
                "arn": "arn:aws:sts::111122223333:user/janedoe",
                "accountId": "111122223333",
                "userName": "janedoe"
            },
            "attributes": {
                "creationDate": "2021-08-02T23:40:45Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2021-08-02T23:40:58Z",
    "eventSource": "redshift.amazonaws.com",
    "eventName": "AuthorizeDataShare",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "3.227.36.75",
    "userAgent":"aws-cli/1.18.118 Python/3.6.10
 Linux/4.9.217-0.1.ac.205.84.332.metal1.x86_64 botocore/1.17.41",
    "requestParameters": {
        "dataShareArn": "arn:aws:redshift:us-
east-1:111122223333:datashare:4c64c6ec-73d5-42be-869b-b7f7c43c7a53/testshare",
        "consumerIdentifier": "555555555555"
    },
    "responseElements": {
        "dataShareArn": "arn:aws:redshift:us-
east-1:111122223333:datashare:4c64c6ec-73d5-42be-869b-b7f7c43c7a53/testshare",
```

```
        "producerNamespaceArn": "arn:aws:redshift:us-
east-1:123456789012:namespace:4c64c6ec-73d5-42be-869b-b7f7c43c7a53",
        "producerArn": "arn:aws:redshift:us-
east-1:111122223333:namespace:4c64c6ec-73d5-42be-869b-b7f7c43c7a53",
        "allowPubliclyAccessibleConsumers": true,
        "dataShareAssociations": [
            {
                "consumerIdentifier": "555555555555",
                "status": "AUTHORIZED",
                "createdDate": "Aug 2, 2021 11:40:56 PM",
                "statusChangeDate": "Aug 2, 2021 11:40:57 PM"
            }
        ]
    },
    "requestID": "87ee1c99-9e41-42be-a5c4-00495f928422",
    "eventID": "03a3d818-37c8-46a6-aad5-0151803bdb09",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


Amazon Redshift Serverless example

Amazon Redshift Serverless is integrated with AWS CloudTrail to provide a record of actions taken in Amazon Redshift Serverless. CloudTrail captures all API calls for Amazon Redshift Serverless as events. For more information about Amazon Redshift Serverless features, see Amazon Redshift Serverless feature overview.

The following example shows a CloudTrail log entry that demonstrates the `CreateNamespace` action.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AAKEOFPINEXAMPLE:admin",
        "arn": "arn:aws:sts::111111111111:assumed-role/admin/admin",
        "accountId": "111111111111",
        "accessKeyId": "AAKEOFPINEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
```

```
                "type": "Role",
                "principalId": "AAKEOFPINEXAMPLE",
                "arn": "arn:aws:iam::111111111111:role/admin",
                "accountId": "111111111111",
                "userName": "admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-03-21T20:51:58Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-03-21T23:15:40Z",
    "eventSource": "redshift-serverless.amazonaws.com",
    "eventName": "CreateNamespace",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "56.23.155.33",
    "userAgent": "aws-cli/2.4.14 Python/3.8.8 Linux/5.4.181-109.354.amzn2int.x86_64
 exe/x86_64.amzn.2 prompt/off command/redshift-serverless.create-namespace",
    "requestParameters": {
        "adminUserPassword": "HIDDEN_DUE_TO_SECURITY_REASONS",
        "adminUsername": "HIDDEN_DUE_TO_SECURITY_REASONS",
        "dbName": "dev",
        "namespaceName": "testnamespace"
    },
    "responseElements": {
        "namespace": {
            "adminUsername": "HIDDEN_DUE_TO_SECURITY_REASONS",
            "creationDate": "Mar 21, 2022 11:15:40 PM",
            "defaultIamRoleArn": "",
            "iamRoles": [],
            "logExports": [],
            "namespaceArn": "arn:aws:redshift-serverless:us-
east-1:111111111111:namespace/befa5123-16c2-4449-afca-1d27cb40fc99",
            "namespaceId": "8b726a0c-16ca-4799-acca-1d27cb403599",
            "namespaceName": "testnamespace",
            "status": "AVAILABLE"
        }
    },
    "requestID": "ed4bb777-8127-4dae-aea3-bac009999163",
    "eventID": "1dbee944-f889-4beb-b228-7ad0f312464",
    "readOnly": false,
    "eventType": "AwsApiCall",
```

```
    "managementEvent": true,
    "recipientAccountId": "111111111111",
    "eventCategory": "Management",
}
```

Amazon Redshift Data API examples

The following example shows a CloudTrail log entry that demonstrates the `ExecuteStatement` action.

```
{
    "eventVersion":"1.05",
    "userIdentity":{
        "type":"IAMUser",
        "principalId":"AKIAIOSFODNN7EXAMPLE:janedoe",
        "arn":"arn:aws:sts::123456789012:user/janedoe",
        "accountId":"123456789012",
        "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
        "userName": "janedoe"
    },
    "eventTime":"2020-08-19T17:55:59Z",
    "eventSource":"redshift-data.amazonaws.com",
    "eventName":"ExecuteStatement",
    "awsRegion":"us-east-1",
    "sourceIPAddress":"192.0.2.0",
    "userAgent":"aws-cli/1.18.118 Python/3.6.10
 Linux/4.9.217-0.1.ac.205.84.332.metal1.x86_64 botocore/1.17.41",
    "requestParameters":{
        "clusterIdentifier":"example-cluster-identifier",
        "database":"example-database-name",
        "dbUser":"example_db_user_name",
        "sql":"***OMITTED***"
    },
    "responseElements":{
        "clusterIdentifier":"example-cluster-identifier",
        "createdAt":"Aug 19, 2020 5:55:58 PM",
        "database":"example-database-name",
        "dbUser":"example_db_user_name",
        "id":"5c52b37b-9e07-40c1-98de-12ccd1419be7"
    },
    "requestID":"00c924d3-652e-4939-8a7a-cd0612eeb8ac",
    "eventID":"c1fb7076-102f-43e5-9ec9-40820bcc1175",
    "readOnly":false,
```

```
    "eventType":"AwsApiCall",
    "recipientAccountId":"123456789012"
}
```

The following example shows a CloudTrail log entry that demonstrates the `ExecuteStatement` action showing the `clientToken` used for idempotency.

```
{
    "eventVersion":"1.05",
    "userIdentity":{
        "type":"IAMUser",
        "principalId":"AKIAIOSFODNN7EXAMPLE:janedoe",
        "arn":"arn:aws:sts::123456789012:user/janedoe",
        "accountId":"123456789012",
        "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
        "userName": "janedoe"
    },
    "eventTime":"2020-08-19T17:55:59Z",
    "eventSource":"redshift-data.amazonaws.com",
    "eventName":"ExecuteStatement",
    "awsRegion":"us-east-1",
    "sourceIPAddress":"192.0.2.0",
    "userAgent":"aws-cli/1.18.118 Python/3.6.10
 Linux/4.9.217-0.1.ac.205.84.332.metal1.x86_64 botocore/1.17.41",
    "requestParameters":{
        "clusterIdentifier":"example-cluster-identifier",
        "database":"example-database-name",
        "dbUser":"example_db_user_name",
        "sql":"***OMITTED***",
        "clientToken":"32db2e10-69ac-4534-b3fc-a191052616ce"
    },
    "responseElements":{
        "clusterIdentifier":"example-cluster-identifier",
        "createdAt":"Aug 19, 2020 5:55:58 PM",
        "database":"example-database-name",
        "dbUser":"example_db_user_name",
        "id":"5c52b37b-9e07-40c1-98de-12ccd1419be7"
    },
    "requestID":"00c924d3-652e-4939-8a7a-cd0612eeb8ac",
    "eventID":"c1fb7076-102f-43e5-9ec9-40820bcc1175",
    "readOnly":false,
    "eventType":"AwsApiCall",
    "recipientAccountId":"123456789012"
```

```
}
```

Amazon Redshift query editor v2 example

The following example shows a CloudTrail log entry that demonstrates the `CreateConnection` action.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AAKEOFPINEXAMPLE:session",
        "arn": "arn:aws:sts::123456789012:assumed-role/MyRole/session",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AAKEOFPINEXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/MyRole",
                "accountId": "123456789012",
                "userName": "MyRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-09-21T17:19:02Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-09-21T22:22:05Z",
    "eventSource": "sqlworkbench.amazonaws.com",
    "eventName": "CreateConnection",
    "awsRegion": "ca-central-1",
    "sourceIPAddress": "192.2.0.2",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:102.0)
 Gecko/20100101 Firefox/102.0",
    "requestParameters": {
        "password": "***",
        "databaseName": "***",
        "isServerless": false,
        "name": "***",
        "host": "redshift-cluster-2.c8robpbxvbf9.ca-central-1.redshift.amazonaws.com",
```

```
        "authenticationType": "***",
        "clusterId": "redshift-cluster-2",
        "username": "***",
        "tags": {
            "sqlworkbench-resource-owner": "AAKEOFPINEXAMPLE:session"
        }
    },
    "responseElements": {
        "result": true,
        "code": "",
        "data": {
            "id": "arn:aws:sqlworkbench:ca-central-1:123456789012:connection/ce56b1be-
dd65-4bfb-8b17-12345123456",
            "name": "***",
            "authenticationType": "***",
            "databaseName": "***",
            "secretArn": "arn:aws:secretsmanager:ca-
central-1:123456789012:secret:sqlworkbench!7da333b4-9a07-4917-b1dc-12345123456-qTCoFm",
            "clusterId": "redshift-cluster-2",
            "dbUser": "***",
            "userSettings": "***",
            "recordDate": "2022-09-21 22:22:05",
            "updatedDate": "2022-09-21 22:22:05",
            "accountId": "123456789012",
            "tags": {
                "sqlworkbench-resource-owner": "AAKEOFPINEXAMPLE:session"
            },
            "isServerless": false
        }
    },
    "requestID": "9b82f483-9c03-4cdd-bb49-a7009e7da714",
    "eventID": "a7cdd442-e92f-46a2-bc82-2325588d41c3",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```


Amazon Redshift account IDs in AWS CloudTrail logs

When Amazon Redshift calls another AWS service for you, the call is logged with an account ID that belongs to Amazon Redshift. It isn't logged with your account ID. For example, suppose

that Amazon Redshift calls AWS Key Management Service (AWS KMS) operations such as `CreateGrant`, `Decrypt`, `Encrypt`, and `RetireGrant` to manage encryption on your cluster. In this case, the calls are logged by AWS CloudTrail using an Amazon Redshift account ID.

Amazon Redshift uses the account IDs in the following table when calling other AWS services.

Region	Region	Account ID
US East (N. Virginia) Region	us-east-1	368064434614
US East (Ohio) Region	us-east-2	790247189693
US West (N. California) Region	us-west-1	703715109447
US West (Oregon) Region	us-west-2	473191095985
Africa (Cape Town) Region	af-south-1	420376844563
Asia Pacific (Hong Kong) Region	ap-east-1	651179539253
Asia Pacific (Hyderabad) Region	ap-south-2	297058826802
Asia Pacific (Jakarta) Region	ap-southeast-3	623197973179
Asia Pacific (Malaysia) Region	ap-southeast-5	590184011157
Asia Pacific (Melbourne) Region	ap-southeast-4	945512339897
Asia Pacific (Mumbai) Region	ap-south-1	408097707231
Asia Pacific (New Zealand)	ap-southeast-6	730335362089
Asia Pacific (Osaka) Region	ap-northeast-3	398671365691
Asia Pacific (Seoul) Region	ap-northeast-2	713597048934
Asia Pacific (Singapore) Region	ap-southeast-1	960118270566
Asia Pacific (Sydney) Region	ap-southeast-2	485979073181
Asia Pacific (Taipei)	ap-east-2	211125526206

Region	Region	Account ID
Asia Pacific (Thailand) Region	ap-southeast-7	767397930036
Asia Pacific (Tokyo) Region	ap-northeast-1	615915377779
Canada (Central) Region	ca-central-1	764870610256
Canada West (Calgary) Region	ca-west-1	830903446466
Europe (Frankfurt) Region	eu-central-1	434091160558
Europe (Ireland) Region	eu-west-1	246478207311
Europe (London) Region	eu-west-2	885798887673
Europe (Milan) Region	eu-south-1	041313461515
Europe (Paris) Region	eu-west-3	694668203235
Europe (Spain) Region	eu-south-2	028811157404
Europe (Stockholm) Region	eu-north-1	553461782468
Europe (Zurich) Region	eu-central-2	668912161003
Israel (Tel Aviv) Region	il-central-1	901883065212
Mexico (Central) Region	mx-central-1	058264411980
Middle East (Bahrain) Region	me-south-1	051362938876
Middle East (UAE) Region	me-central-1	595013617770
South America (São Paulo) Region	sa-east-1	392442076723

The following example shows a CloudTrail log entry for the AWS KMS Decrypt operation that was called by Amazon Redshift.

```
{
```

```
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAI5QPCMKLTL4VHFCYY:i-0f53e22dbe5df8a89",
        "arn": "arn:aws:sts::790247189693:assumed-role/prod-23264-role-wp/
i-0f53e22dbe5df8a89",
        "accountId": "790247189693",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2017-03-03T16:24:54Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAI5QPCMKLTL4VHFCYY",
                "arn": "arn:aws:iam::790247189693:role/prod-23264-role-wp",
                "accountId": "790247189693",
                "userName": "prod-23264-role-wp"
            }
        }
    },
    "eventTime": "2017-03-03T17:16:51Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "52.14.143.61",
    "userAgent": "aws-internal/3",
    "requestParameters": {
        "encryptionContext": {
            "aws:redshift:createtime": "20170303T1710Z",
            "aws:redshift:arn": "arn:aws:redshift:us-east-2:123456789012:cluster:my-dw-
instance-2"
        }
    },
    "responseElements": null,
    "requestID": "30d2fe51-0035-11e7-ab67-17595a8411c8",
    "eventID": "619bad54-1764-4de4-a786-8898b0a7f40c",
    "readOnly": true,
    "resources": [
        {
            "ARN": "arn:aws:kms:us-east-2:123456789012:key/f8f4f94f-e588-4254-
b7e8-078b99270be7",
```

```
            "accountId": "123456789012",
            "type": "AWS::KMS::Key"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012",
    "sharedEventID": "c1daefea-a5c2-4fab-b6f4-d8eaa1e522dc"

}
```

System table integration with S3 Tables

Use system table integration with Amazon S3 Tables to retain your Amazon Redshift system table data beyond the 7-day in-cluster retention period. This helps you meet compliance, auditing, and observability requirements. When you enable this feature, Redshift writes system table logs to S3 Tables in Apache Iceberg format and manages partitioning, compaction, and snapshot maintenance. You can keep this data for as long as you need for analysis and auditing. Because the data is stored in open Iceberg format, you can query it with Redshift, Amazon Athena, or any other Iceberg-compatible engine. You can also use AI agent skills for natural-language queries (see System table skills for AWS Agent Toolkit on the GitHub website).

When you enable this feature through the console, the Redshift CLI, or SDK, Redshift begins writing new rows from the system tables you select to S3 Tables shortly thereafter at a fixed frequency. You choose which system tables to publish; S3 Tables enforces the retention duration that you configure.

This supports the following use cases:

- **Meet compliance and audit requirements**. Retain query, connection, and change history for months or years.
- **Monitor your fleet from one place**. Consolidate monitoring data from multiple data warehouses in your account and Region and analyze activity across your fleet.
- **Remove the need for custom extract, transform, and load (ETL) pipelines**. Stop building and maintaining jobs that export system table data to S3 for longer retention.
- **Investigate past incidents**. Query historical data beyond the 7-day in-cluster window for root cause analysis, post-incident review, and monitoring workload evolution and performance trends over time.

This feature is available for Amazon Redshift Provisioned RA3 and RG instances and Amazon Redshift Serverless in all AWS commercial Regions where Redshift and S3 Tables are supported. In this topic, *data warehouse* refers to either a provisioned cluster or a Redshift Serverless workgroup.

Topics

- How it works
- Enabling system table integration with S3 Tables
- Supported system tables
- Metadata columns added to each table
- Configuring retention
- Data governance
- Delivery and lifecycle behavior
- Best practices
- Billing
- Considerations and limitations

How it works

After you enable the feature and select one or more supported system tables, Redshift writes newly completed records from those system tables to S3 Tables. Data is delivered in batches at a fixed frequency.

The system tables live in your AWS account but are *service-managed*: AWS owns and operates the data delivery and table management, including the following:

- Creating the S3 table bucket, namespace, and S3 tables in your account.
- Defining and evolving the table schemas.
- Writing data in Apache Iceberg format to S3 tables.
- Compacting data and maintaining snapshots (handled automatically by S3 Tables).

There is no infrastructure to build or maintain and no impact to your workloads.

You configure how long to retain the data through S3 Tables record expiration. For more information, see the section called “Configuring retention”.

Because the tables are managed by AWS, you cannot modify or delete the rows that Redshift delivers. The delivered data is immutable. You only control read access to the tables. For more information, see the section called “Data governance”.

Only completed activity is delivered to S3 Tables. The system tables in your data warehouse can show in-flight activity, but only records that have reached a final state are copied. For example, a completed, aborted, or canceled query is delivered, but a query that is still running is not delivered until it reaches a final state.

Enabling system table integration with S3 Tables

To enable system table integration with S3 Tables, use the existing Redshift APIs with a new log destination type of `s3table`. When enabled, Redshift creates an S3 table bucket named `aws-redshift` in your account. You manage the feature with the same operations you use for other log destinations: for provisioned clusters, `enable-logging`, `disable-logging`, and `describe-logging-status`; for Redshift Serverless, `update-namespace` and `get-namespace`.

Permissions

The principal that enables, modifies, or disables the feature must have the following permissions:

- `redshift:EnableLogging` (for provisioned clusters) or `redshift-serverless:UpdateNamespace` (for Redshift Serverless).
- Permission to create and configure the S3 table bucket. The enabling principal needs the following S3 Tables permissions:
  - `s3tables:CreateTableBucket`
  - `s3tables:PutTableBucketEncryption`
  - `s3tables:PutTableBucketPolicy`

  When you enable the feature, Redshift creates and configures the S3 table bucket using the identity of the principal who triggers the operation.

After the bucket is created, Redshift creates namespaces and tables within it using a service trust relationship between Redshift and S3 Tables. The enabling principal does not need permissions to create namespaces or tables.

Deployment options

You can configure delivery in one of two patterns. A given data warehouse uses one pattern at a time.

Deployment Option	Description
Per-warehouse	Redshift writes each data warehouse's system table data to its own set of S3 tables. Every row in a table comes from a single data warehouse. Use this for physical isolation between warehouses.
Consolidated	Redshift writes system table data from multiple data warehouses within the same account and Region into a shared set of S3 tables. Rows from different warehouses are distinguished by the `warehouse_namespace_arn` and `warehouse_name` columns. Use this for centralized, cross-warehouse analysis.

The deployment model determines how data is organized in S3 Tables. In the per-warehouse model, the S3 Tables namespace name contains a unique identifier for the data warehouse, so each warehouse's data is physically separated into its own namespace. In the consolidated model, the S3 Tables namespace name contains the AWS account number, and data from all warehouses in that account and Region is stored together in the same namespace and tables.

Both options are supported within a **single AWS account and a single AWS Region**. To analyze data across Regions or accounts, combine results from each Region's or account's tables at query time. For cross-account access, you can use AWS Glue Data Catalog sharing. For more information, see Granting cross-account access in the *AWS Glue Developer Guide*.

> **Note**
>
> If you switch a data warehouse from one deployment pattern to another, previously delivered data remains in its existing table and is retained according to its configured retention. Redshift begins delivering new data to the new target and does not backfill historical data.

Using the console

To configure system table integration with S3 Tables

1. Open the Redshift console and in the navigation pane, under **Amazon Redshift** or **Amazon Redshift Serverless**, choose **System table integrations**. You can also configure from the cluster or namespace detail page by choosing the **Integrations** tab, or by choosing **Actions**, **Integrations**, **Configure system table integration**.
2. Choose **Create System table integration**.
3. Select your provisioned cluster or Redshift Serverless workgroup.
4. Choose the system tables to publish. Select individual SYS_* views, or choose **Select all supported system tables** to publish all current and future supported views. If you select all, new views added in the future are automatically included without requiring a configuration change.
5. Select the deployment model:
   - **Individual S3 table per system table per data warehouse** to keep this warehouse's data in its own set of tables.
   - **Shared S3 table per system table across data warehouses** to consolidate data from multiple warehouses in the account into a shared set of tables.
6. Optionally, choose an encryption key. By default the data is encrypted with Amazon S3-managed keys (SSE-S3). To use your own AWS KMS key, select it here. If you use your own AWS KMS key, the key policy must grant Redshift and S3 Tables access to the key. For more information, see the section called “Configuring encryption”.
7. Save your changes. Redshift begins publishing the selected system tables to S3 Tables and continues to add new records on a recurring basis.

To stop publishing a system table, return to the settings and remove it. Data already published is retained until you configure expiration through S3 Tables.

You can review the status of the integration from the cluster or namespace detail page. Expand **View S3 table mapping** to see the mapping between each system table and its corresponding S3 table name and namespace. You can also view the published data from the S3 Tables console.

Using the AWS CLI

The following parameters configure S3 Tables publishing:

Parameter	Applies to	Description
`--log-destination-type`	Both	Set to `s3table` to publish to S3 Tables. Other values are `s3` and `cloudwatch` .
`--log-exports`	Provisioned	The system tables to publish, as a list of SYS_* view names, or `all`.
`--s3-table-names`	Serverless	The system tables to publish, as a list of SYS_* view names, or `all`.
`--s3-table-action`	Serverless	`Enable` or `Disable` S3 Tables publishing.
`--s3-table-granularity`	Both	Table scope. Provisioned: `cluster` (default) or `account`. Serverless: `namespace` (default) or `account`.
`--s3-table-kms-key-id`	Both	Optional AWS KMS key ARN or ID for encryption. Defaults to Amazon S3-managed keys (SSE-S3).

Provisioned clusters

Enable delivery. `--log-exports` accepts `all` or a space-separated list of supported system tables, and `--s3-table-granularity` accepts `cluster` (per-warehouse) or `account` (consolidated).

```
aws redshift enable-logging \
    --cluster-identifier my-redshift-cluster \
    --log-destination-type s3table \
    --log-exports all \
    --s3-table-granularity account
```

To enable delivery for specific system tables only:

```
aws redshift enable-logging \
    --cluster-identifier my-redshift-cluster \
    --log-destination-type s3table \
    --log-exports sys_query_history sys_query_text sys_userlog \
    --s3-table-granularity cluster
```

Check delivery status.

```
aws redshift describe-logging-status \
    --cluster-identifier my-redshift-cluster
```

Disable delivery for all system tables, or for specific tables with `--log-exports`:

```
aws redshift disable-logging \
    --cluster-identifier my-redshift-cluster \
    --log-destination-type s3table \
    --log-exports sys_stream_scan_states
```

Redshift Serverless

In Redshift Serverless, you configure the feature per namespace with `update-namespace`. System table names are passed on `--s3-table-names` (not `--log-exports`).

Enable delivery.

```
aws redshift-serverless update-namespace \
    --namespace-name my-namespace \
    --log-destination-type s3table \
    --s3-table-action Enable \
    --s3-table-names all \
    --s3-table-granularity namespace
```

Disable delivery for specific system tables (or `all`):

```
aws redshift-serverless update-namespace \
    --namespace-name my-namespace \
    --log-destination-type s3table \
    --s3-table-action Disable \
    --s3-table-names sys_stream_scan_states
```

Check delivery status.

```
aws redshift-serverless get-namespace \
    --namespace-name my-namespace
```

To resume delivery after the S3 table bucket or a table was dropped, run the enable command again.

Registering with AWS Glue Data Catalog

Before you can query the retained data with Redshift, Athena, or other analytics services, the service-managed S3 table bucket must be integrated with AWS Glue Data Catalog. This is a one-time step per account and Region. If you have already integrated S3 Tables with AWS Glue Data Catalog, no additional action is required.

For integration instructions, see Integrating Amazon S3 Tables with AWS Glue Data Catalog in the *AWS Glue Developer Guide*.

Configuring encryption

By default, data delivered to S3 Tables is encrypted with Amazon S3-managed keys (SSE-S3). To use an AWS KMS customer managed key instead, specify its ARN with `--s3-table-kms-key-id` when you enable delivery:

```
aws redshift enable-logging \
    --cluster-identifier my-redshift-cluster \
    --log-destination-type s3table \
    --log-exports all \
    --s3-table-granularity account \
    --s3-table-kms-key-id arn:aws:kms:us-west-2:111122223333:key/key-id
```

If you use a customer managed key, its key policy must allow the Redshift system table integration service principal to generate data keys when it writes your system tables, and the S3 Tables maintenance service principal to use the key during table maintenance (such as compaction). Add the following statements to the key policy, replacing *PARTITION*, *REGION*, *ACCOUNT*, and *KEY_ID* with your own values.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnableRedshiftSystemTableKeyUsage",
      "Effect": "Allow",
      "Principal": {
        "Service": "systemtables.redshift.amazonaws.com"
      },
      "Action": [
        "kms:DescribeKey",
        "kms:GenerateDataKey",
```

```
        "kms:Decrypt"
      ],
      "Resource": "arn:PARTITION:kms:REGION:ACCOUNT:key/KEY_ID",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "ACCOUNT"
        }
      }
    },
    {
      "Sid": "EnableS3TableMaintenanceKeyUsage",
      "Effect": "Allow",
      "Principal": {
        "Service": "maintenance.s3tables.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "arn:PARTITION:kms:REGION:ACCOUNT:key/KEY_ID",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws:s3:arn":
 "arn:PARTITION:s3tables:REGION:ACCOUNT:bucket/aws-redshift/*"
        }
      }
    }
  ]
}
```

If the key policy does not grant these permissions, Redshift cannot write to the encrypted S3 Tables and delivery fails.

Supported system tables

You can select any of the following SYS_* monitoring views to integrate with S3 Tables. For a description of the columns in each view, choose the view name.

- SYS_ANALYZE_COMPRESSION_HISTORY
- SYS_ANALYZE_HISTORY
- SYS_AUTOMATIC_OPTIMIZATION
- SYS_AUTO_TABLE_OPTIMIZATION

- SYS_CHILD_QUERY_TEXT
- SYS_CONNECTION_LOG
- SYS_COPY_JOB_INFO
- SYS_COPY_REPLACEMENTS
- SYS_DATASHARE_CHANGE_LOG
- SYS_DATASHARE_USAGE_CONSUMER
- SYS_DATASHARE_USAGE_PRODUCER
- SYS_DATASHARE_WRITE_HISTORY
- SYS_EXTERNAL_QUERY_ERROR
- SYS_INTEGRATION_ACTIVITY
- SYS_MV_STATE
- SYS_PROCEDURE_MESSAGES
- SYS_QUERY_DETAIL
- SYS_QUERY_EXPLAIN
- SYS_QUERY_HISTORY
- SYS_QUERY_TEXT
- SYS_SCHEMA_QUOTA_VIOLATIONS
- SYS_SESSION_HISTORY
- SYS_SPATIAL_SIMPLIFY
- SYS_STREAM_SCAN_ERRORS
- SYS_STREAM_SCAN_STATES
- SYS_UNLOAD_DETAIL
- SYS_USERLOG
- SYS_VACUUM_HISTORY

Note

The following views require patch P203 or later: SYS_CHILD_QUERY_TEXT, SYS_COPY_REPLACEMENTS, SYS_EXTERNAL_QUERY_ERROR, SYS_PROCEDURE_MESSAGES, SYS_QUERY_DETAIL, SYS_QUERY_EXPLAIN,

SYS_SPATIAL_SIMPLIFY, and SYS_UNLOAD_DETAIL. If you enable these views on a data warehouse running a patch earlier than P203, the tables are created with the correct schema but contain no data until the data warehouse is updated to P203 or later.

Metadata columns added to each table

Each S3 table contains all of the columns from the source SYS_* view, plus the following five metadata columns that Redshift adds to identify the source of each row and when it was delivered.

Column	Type	Description
`warehouse_account_id`	string	The AWS account that owns the source data warehouse.
`warehouse_region_name`	string	The AWS Region in which the source data warehouse runs.
`warehouse_namespace_arn`	string	The ARN of the namespace for the source data warehouse. This is a stable, unique identifier that is immutable across renames and recreation.
`warehouse_name`	string	The human-readable name of the source data warehouse (cluster name or workgroup name).
`s3_tables_ingestion_time`	timestamp(6), UTC	The time at which Redshift committed the row to S3 Tables. This is the delivery time, not the time the underlying event occurred.

Configuring retention

Redshift does not define a retention policy for the data it delivers to S3 Tables. You configure how long to keep the data by setting a record expiration policy through S3 Tables. You specify the number of days (for example, 5 days, 100 days, or 365 days), and S3 Tables automatically removes data that ages past the configured duration.

If you do not configure an expiration policy, data is kept indefinitely.

When you disable the integration or remove a system table, Redshift stops writing new data but previously delivered data remains in S3 Tables and continues to be subject to whatever expiration policy you have configured.

For more information, see Configuring record expiration for S3 table buckets in the *Amazon S3 User Guide*.

Data governance

You govern access to the retained system table data; Redshift does not apply read permissions on your behalf and does not mask data.

- **Access control**. After you integrate the S3 table bucket with AWS Glue Data Catalog, you can govern read access using AWS Identity and Access Management (IAM) or AWS Lake Formation. In a consolidated deployment, you can scope read access per warehouse.
- **Cataloging and discovery**. The AWS Glue Data Catalog integration provides a central place to discover the retained tables and apply fine-grained permissions (for example, table-, column-, or row-level access with Lake Formation).

Delivery and lifecycle behavior

The following behaviors apply to data delivery and the lifecycle of the integration:

- **Delivery frequency**. Data is delivered in batches at a fixed frequency.
- **Exactly-once delivery**. Each record is delivered once; re-enabling or re-adding a system table does not create duplicates.
- **Enable, disable, re-enable**. While the feature (or a specific system table) is disabled, no data is captured for that period. Disabling stops writes but does not remove the S3 Tables or previously delivered data. Re-enabling resumes delivery going forward and does not backfill the gap.
- **Removing a system table from your selection**. Redshift stops delivering new data for that system table. Previously delivered data remains in S3 Tables and is subject to any record expiration policy you have configured. Re-adding the system table resumes delivery without duplicating data.
- **Data warehouse deletion with in-flight data**. If you delete a data warehouse while delivery is in progress and the data warehouse's encryption key differs from the key used for S3 Tables, Redshift might continue to use the AWS KMS grant on the data warehouse's key for a short period after deletion to complete delivery of any remaining data.

Checking delivery status

You can review the current configuration and the most recent delivery time per system table at any time:

- **Provisioned clusters**. `describe-logging-status` (or `describe-cluster`) returns the active system tables, the S3 Tables namespace, the granularity, and the last ingestion time for each system table.
- **Redshift Serverless**. `get-namespace` returns the same information in the namespace's S3 Tables publish status.

Redshift Serverless

Delivery does not keep a workgroup awake or consume your compute. Before the workgroup pauses, Redshift ensures any pending batch is delivered.

Best practices

1. **Assess your data sensitivity.** Understand what kind of data is stored within your data warehouse and whether the system tables contain sensitive information. Some system tables (such as SYS_QUERY_TEXT and SYS_PROCEDURE_MESSAGES) can capture literal values from your queries and stored procedures.
2. **Choose a deployment model.** If your data warehouse stores sensitive information, consider using the per-warehouse deployment model to keep each data warehouse's system table data physically isolated. If your data is not sensitive and you want to run cross-warehouse queries without combining results from multiple tables, use the consolidated deployment model.
3. **Select the system tables you need.** Review the list of supported system tables and choose the ones that match your compliance, audit, or observability requirements. You do not need to enable all system tables.
4. **Configure encryption before you enable the feature.** If you want to use a customer managed AWS KMS key, specify it when you first enable the feature. You cannot change the key after S3 Tables are created. To change the key, disable the feature, drop the S3 Tables using the S3 Tables API (this permanently deletes retained data), and then re-enable the feature with the new key.
5. **Set retention based on business requirements.** Configure the record expiration policy directly in S3 Tables at a table level, based on your compliance, audit, or operational needs. Different tables can have different retention durations. If you do not configure an expiration policy, data

is kept indefinitely. To monitor storage usage for your system tables, see Amazon S3 Tables CloudWatch metrics in the *Amazon S3 User Guide*.

Billing

Writing system table data to S3 Tables is free. You are charged for standard S3 Tables storage and maintenance for the retained data, and for the query engine you use to read the data, according to that engine's pricing. For more information, see Amazon S3 Tables pricing.

Considerations and limitations

- Supported within a single AWS account and a single AWS Region. To analyze data across accounts or Regions, combine results at query time.
- A data warehouse uses one deployment pattern (per-warehouse or consolidated) at a time.
- Switching deployment patterns, disabling and re-enabling, or removing and re-adding a system table does not backfill historical data.
- Delivered data is immutable. You cannot modify or delete individual rows through Redshift.
- You can drop the S3 Tables created by this feature, but doing so permanently removes all retained data in those tables and stops delivery. Redshift does not automatically recreate dropped tables. To resume delivery, you must re-enable the feature, which creates new tables and begins delivering new data going forward. Previously delivered data is not restored. For more information, see Deleting S3 tables in the *Amazon S3 User Guide*.
- Delivery is batch-based (data is written at a fixed frequency).
- Querying from Redshift requires the S3 table bucket to be integrated with AWS Glue Data Catalog.

Compliance validation for Amazon Redshift

Third-party auditors assess the security and compliance of Amazon Redshift as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS services in scope by compliance program. For general information, see AWS compliance programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading reports in AWS Artifact.

Your compliance responsibility when using Amazon Redshift is determined by the sensitivity of your data, your organization's compliance objectives, and applicable laws and regulations. If your use of Amazon Redshift is subject to compliance with standards like HIPAA, PCI, or FedRAMP, AWS provides resources to help:

- Security and compliance quick start guides that discuss architectural considerations and steps for deploying security- and compliance- focused baseline environments on AWS.
- Architecting for HIPAA security and compliance whitepaper, which describes how companies can use AWS to create HIPAA-compliant applications.
- AWS compliance resources, workbooks and guides that might apply to your industry and location.
- AWS Config, an AWS service, can assess how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub CSPM, an AWS service, provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices. Security Hub CSPM uses security controls to evaluate resource configurations and security standards to help you comply with various compliance frameworks. For more information about using Security Hub CSPM to evaluate Amazon Redshift resources, see Amazon Redshift controls in the *AWS Security Hub User Guide*.

The following compliance and security documents cover Amazon Redshift and are available on demand through AWS Artifact. For more information, see AWS Artifact.

- Cloud Computing Compliance Controls Catalogue (C5)
- ISO 27001:2013 Statement of Applicability (SoA)
- ISO 27001:2013 Certification
- ISO 27017:2015 Statement of Applicability (SoA)
- ISO 27017:2015 Certification
- ISO 27018:2015 Statement of Applicability (SoA)
- ISO 27018:2014 Certification
- ISO 9001:2015 Certification
- PCI DSS Attestation of Compliance (AOC) and Responsibility Summary
- Service Organization Controls (SOC) 1 Report
- Service Organization Controls (SOC) 2 Report

- Service Organization Controls (SOC) 2 Report For Confidentiality

Resilience in Amazon Redshift

The AWS global infrastructure is built around AWS Regions and Availability Zones (AZs). AWS Regions provide multiple, physically separated and isolated Availability Zones that are connected with low latency, high throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single data center infrastructures or multiple data center infrastructures.

Almost all AWS Regions have multiple Availability Zones and data centers. You can deploy your applications across multiple Availability Zones in the same Region for fault tolerance and low latency.

To move a cluster to another Availability Zone without any loss of data or changes to your applications, you can set up relocation for your cluster. With relocation, you can continue operations when there is an interruption of service on your cluster with minimal impact. When cluster relocation is turned on, Amazon Redshift might choose to relocate clusters in some situations. For more information on relocation in Amazon Redshift, see Relocating a cluster.

In failure scenarios where an unexpected event happens in an Availability Zone, you can set up a multiple Availability Zones (Multi-AZ) deployment to ensure that your Amazon Redshift data warehouse can continue operating. Amazon Redshift deploys equal compute resources in two Availability Zones that can be accessed through a single endpoint. In the event of an entire Availability Zone failure, the remaining compute resources in the second Availability Zone will be available to continue processing workloads. For more information on Multi-AZ deployments, see Multi-AZ deployment.

For more information on AWS Regions and Availability Zones, see AWS global infrastructure. For more information on using Amazon Redshift for disaster recovery, see Implement disaster recovery with Amazon Redshift.

.

Infrastructure security in Amazon Redshift

As a managed service, Amazon Redshift is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access Amazon Redshift through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Network isolation

A virtual private cloud (VPC) based on the Amazon VPC service is your private, logically isolated network in the AWS Cloud. You can deploy an Amazon Redshift cluster or Redshift Serverless workgroup within a VPC by taking the following steps:

- Create a VPC in an AWS Region. For more information, see What is Amazon VPC? in the *Amazon VPC User Guide.*
- Create two or more private VPC subnets. For more information, see VPCs and subnets in the *Amazon VPC User Guide.*
- Deploy an Amazon Redshift cluster or a Redshift Serverless workgroup. For more information, see Subnets for Redshift resources or Workgroups and namespaces.

An Amazon Redshift cluster is locked down by default upon provisioning. To allow inbound network traffic from Amazon Redshift clients, associate a VPC security group with an Amazon Redshift cluster. For more information, see Subnets for Redshift resources.

To allow traffic only to or from specific IP address ranges, update the security groups with your VPC. An example is allowing traffic only from or to your corporate network.

While configuring network access control lists associated with the subnet(s) your Amazon Redshift cluster is tagged with, ensure that the respective AWS Region's S3 CIDR ranges are added to the

allowlist for both ingress and egress rules. Doing so lets you execute S3-based operations such as Redshift Spectrum, COPY, and UNLOAD without any disruptions.

The following example command parses the JSON response for all IPv4 addresses used in Amazon S3 in the us-east-1 Region.

```
curl https://ip-ranges.amazonaws.com/ip-ranges.json | jq -r '.prefixes[] |
 select(.region=="us-east-1") | select(.service=="S3") | .ip_prefix'

54.231.0.0/17

52.92.16.0/20

52.216.0.0/15
```

For instructions on how to get S3 IP ranges for a particular region, see AWS IP address ranges.

Amazon Redshift supports deploying clusters into dedicated tenancy VPCs. For more information, see Dedicated instances in the *Amazon EC2 User Guide.*

Amazon Redshift security groups

When you provision an Amazon Redshift cluster, it is locked down by default so nobody has access to it. To grant other users inbound access to an Amazon Redshift cluster, you associate the cluster with a security group. If you are on the EC2-VPC platform, you can either use an existing Amazon VPC security group or define a new one and then associate it with a cluster. For more information on managing a cluster on the EC2-VPC platform, see Redshift resources in a VPC.

Interface VPC endpoints

You can connect directly to the Amazon Redshift and Amazon Redshift Serverless API services using an interface VPC endpoint (AWS PrivateLink) in your virtual private cloud (VPC) instead of connecting over the internet. For information about Amazon Redshift API actions, see Actions in the *Amazon Redshift API Reference*. For information about Redshift Serverless API actions, see Actions in the *Amazon Redshift Serverless API Reference*. For more information about AWS PrivateLink, see Interface VPC endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*. Note that JDBC/ODBC connection to the cluster or workspace is not part of Amazon Redshift API service.

When you use an interface VPC endpoint, communication between your VPC and Amazon Redshift or Redshift Serverless is conducted entirely within the AWS network, which can provide greater

security. Each VPC endpoint is represented by one or more elastic network interfaces with private IP addresses in your VPC subnets. For more information on elastic network interfaces, see Elastic network interfaces in the *Amazon EC2 User Guide.*

An interface VPC endpoint connects your VPC directly to Amazon Redshift. It doesn't use an internet gateway, network address translation (NAT) device, virtual private network (VPN) connection, or Direct Connect connection. The instances in your VPC don't need public IP addresses to communicate with the Amazon Redshift API.

To use Amazon Redshift or Redshift Serverless through your VPC, you have two options. One is to connect from an instance that is inside your VPC. The other is to connect your private network to your VPC by using an Site-to-Site VPN option or Direct Connect. For more information about Site-to-Site VPN options, see VPN connections in the *Amazon VPC User Guide*. For information about Direct Connect, see Creating a Connection in the *Direct Connect User Guide*.

You can create an interface VPC endpoint to connect to Amazon Redshift using the AWS Management Console or AWS Command Line Interface (AWS CLI) commands. For more information, see Creating an Interface Endpoint.

After you create an interface VPC endpoint, you can enable private DNS host names for the endpoint. When you do, the default endpoint is as follows:

- **Amazon Redshift provisioned**: `https://redshift.Region.amazonaws.com`
- **Amazon Redshift Serverless**: `https://redshift-serverless.Region.amazonaws.com`

If you don't enable private DNS host names, Amazon VPC provides a DNS endpoint name that you can use in the following format.

- **Amazon Redshift provisioned**: `VPC_endpoint_ID.redshift.Region.vpce.amazonaws.com`
- **Amazon Redshift Serverless**: `VPC_endpoint_ID.redshift-serverless.Region.vpce.amazonaws.com`

For more information, see Interface VPC endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*.

Amazon Redshift and Redshift Serverless support making calls to all of the Amazon Redshift API operations and Redshift Serverless API operations inside your VPC.

You can attach VPC endpoint policies to a VPC endpoint to control access for AWS Identity and Access Management (IAM) principals. You can also associate security groups with a VPC endpoint to control inbound and outbound access based on the origin and destination of network traffic. An example is a range of IP addresses. For more information, see Controlling Access to Services with VPC Endpoints in the *Amazon VPC User Guide*.

VPC endpoint policies for Amazon Redshift

You can create a policy for VPC endpoints for Amazon Redshift to specify the following:

- The principal that can or can't perform actions
- The actions that can be performed
- The resources on which actions can be performed

For more information, see Controlling access to services with VPC endpoints in the *Amazon VPC User Guide*.

Following, you can find examples of VPC endpoint policies.

Amazon Redshift Provisioned Endpoint Policy Examples

Following, you can find examples of VPC endpoint policies for Amazon Redshift Provisioned.

Example: VPC endpoint policy to deny all access from a specified AWS account

The following VPC endpoint policy denies the AWS account *123456789012* all access to resources using this endpoint.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "*",
```

```
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example: VPC endpoint policy to allow VPC access only to a specified IAM role

The following VPC endpoint policy allows full access only to the IAM role *redshiftrole* in AWS account *123456789012*. All other IAM principals are denied access using the endpoint.

```
   {
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:role/redshiftrole"
                ]
            }
        }]
}
```

This is only a sample. In most use cases we recommend attaching permissions for specific actions to narrow the scope of permissions.

Example: VPC endpoint policy to allow VPC access only to a specified IAM principal (user)

The following VPC endpoint policy allows full access only to the IAM user *redshiftadmin* in AWS account *123456789012*. All other IAM principals are denied access using the endpoint.

```
   {
    "Statement": [
        {
```

```
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:user/redshiftadmin"
                ]
            }
        }]
}
```

This is only a sample. In most use cases we recommend attaching permissions to a role before assigning to a user. Additionally, we recommend using specific actions to narrow the scope of permissions.

Example: VPC endpoint policy to allow read-only Amazon Redshift operations

The following VPC endpoint policy allows only AWS account *123456789012* to perform the specified Amazon Redshift actions.

The actions specified provide the equivalent of read-only access for Amazon Redshift. All other actions on the VPC are denied for the specified account. Also, all other accounts are denied any access. For a list of Amazon Redshift actions, see Actions, Resources, and Condition Keys for Amazon Redshift in the *IAM User Guide.*

```
{
  "Statement": [
      {
          "Action": [
              "redshift:DescribeAccountAttributes",
              "redshift:DescribeClusterParameterGroups",
              "redshift:DescribeClusterParameters",
              "redshift:DescribeClusterSecurityGroups",
              "redshift:DescribeClusterSnapshots",
              "redshift:DescribeClusterSubnetGroups",
              "redshift:DescribeClusterVersions",
              "redshift:DescribeDefaultClusterParameters",
              "redshift:DescribeEventCategories",
              "redshift:DescribeEventSubscriptions",
              "redshift:DescribeHsmClientCertificates",
              "redshift:DescribeHsmConfigurations",
```

```
                "redshift:DescribeLoggingStatus",
                "redshift:DescribeOrderableClusterOptions",
                "redshift:DescribeQuery",
                "redshift:DescribeReservedNodeOfferings",
                "redshift:DescribeReservedNodes",
                "redshift:DescribeResize",
                "redshift:DescribeSavedQueries",
                "redshift:DescribeScheduledActions",
                "redshift:DescribeSnapshotCopyGrants",
                "redshift:DescribeSnapshotSchedules",
                "redshift:DescribeStorage",
                "redshift:DescribeTable",
                "redshift:DescribeTableRestoreStatus",
                "redshift:DescribeTags",
                "redshift:FetchResults",
                "redshift:GetReservedNodeExchangeOfferings"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example: VPC endpoint policy denying access to a specified cluster

The following VPC endpoint policy allows full access for all accounts and principals. At the same time, it denies any access for AWS account *123456789012* to actions performed on the Amazon Redshift cluster with cluster ID *my-redshift-cluster*. Other Amazon Redshift actions that don't support resource-level permissions for clusters are still allowed. For a list of Amazon Redshift actions and their corresponding resource type, see Actions, Resources, and Condition Keys for Amazon Redshift in the *IAM User Guide.*

```
{
   "Statement": [
       {
           "Action": "*",
```

```
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "arn:aws:redshift:us-east-1:123456789012:cluster:my-redshift-
cluster",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Amazon Redshift Serverless Endpoint Policy Examples

Following, you can find examples of VPC endpoint policies for Redshift Serverless.

Example: VPC endpoint policy to allow read-only Redshift Serverless operations

The following VPC endpoint policy allows only AWS account *123456789012* to perform the specified Redshift Serverless actions.

The actions specified provide the equivalent of read-only access for Redshift Serverless. All other actions on the VPC are denied for the specified account. Also, all other accounts are denied any access. For a list of Redshift Serverless actions, see Actions, Resources, and Condition Keys for Redshift Serverless in the *IAM User Guide.*

```
{
  "Statement": [
      {
          "Action": [
              "redshift-serverless:DescribeOneTimeCredit",
              "redshift-serverless:GetCustomDomainAssociation",
              "redshift-serverless:GetEndpointAccess",
              "redshift-serverless:GetNamespace",
```

```
                "redshift-serverless:GetRecoveryPoint",
                "redshift-serverless:GetResourcePolicy",
                "redshift-serverless:GetScheduledAction",
                "redshift-serverless:GetSnapshot",
                "redshift-serverless:GetTableRestoreStatus",
                "redshift-serverless:GetUsageLimit",
                "redshift-serverless:GetWorkgroup"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example: VPC endpoint policy denying access to a specified workgroup

The following VPC endpoint policy allows full access for all accounts and principals. At the same time, it denies any access for AWS account *123456789012* to actions performed on the Amazon Redshift workgroup with workgroup ID *my-redshift-workgroup*. Other Amazon Redshift actions that don't support resource-level permissions for workgroups are still allowed. For a list of Redshift Serverless actions and their corresponding resource type, see Actions, Resources, and Condition Keys for Redshift Serverless in the *IAM User Guide.*

```
 {
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "arn:aws:redshift-serverless:us-
east-1:123456789012:workgroup:my-redshift-workgroup",
```

```
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Configuration and vulnerability analysis in Amazon Redshift

AWS handles basic security tasks like guest operating system (OS) and database patching, firewall configuration, and disaster recovery (DR). These procedures have been reviewed by certified third parties. For more information, see Compliance validation for Amazon Redshift, the Shared responsibility model, and Best Practices for Security, Identity, and Compliance.

Amazon Redshift automatically applies upgrades and patches your data warehouse so you can focus on your application and not on its administration. Patches and upgrades are applied during a configurable maintenance window. For more information, see Maintenance windows.

Amazon Redshift query editor v2 is an AWS-managed application. All patches and updates are applied by AWS as needed.

Connect to Redshift with Identity-enhanced IAM role sessions

You can use IAM Identity Center to provide federated access to your Amazon Redshift clusters and serverless workgroups. This approach allows users to authenticate using their Identity Center credentials.

Amazon Redshift provides `GetIdentityCenterAuthToken` API operations to generate authorized token that contains user identity information. These APIs are available for both provisioned clusters and serverless workgroups. The tokens enable seamless single sign-on access to Amazon Redshift databases using your existing Identity Center setup.

Prerequisites

Before using Identity Center authentication with Amazon Redshift, ensure you have the following:

- **Identity Center setup:** Your account must have IAM Identity Center configured with user identities and appropriate application assignments. For setup instructions, see Setting up IAM Identity Center.

  > **Important**
  >
  > If you want to connect to Redshift, you must use redshift:connect scope.

- **Identity-enhanced credentials:** Your application must use identity-enhanced credentials that contain embedded user identity information. For more information, see Using identity-enhanced IAM role sessions.
- **IAM permissions:** Your IAM role or user must have permissions to call the `GetIdentityCenterAuthToken` API and access the specified clusters or workgroups. Required permissions:
  - For provisioned clusters: `redshift:GetIdentityCenterAuthToken` on cluster ARNs (format: `arn:aws:redshift:region:account:cluster:cluster-name`)
  - For serverless workgroups: `redshift-serverless:GetIdentityCenterAuthToken` on workgroup ARNs (format: `arn:aws:redshift-serverless:region:account:workgroup/workgroup-name`)

- **Compatible drivers:** Use Amazon Redshift JDBC or ODBC drivers that support Identity Center authorized tokens:
  - JDBC drivers: See Installing and configuring the Amazon Redshift JDBC driver version 2.x
  - ODBC drivers: See Installing and configuring the Amazon Redshift ODBC driver version 2.0
- **Network configuration for enhanced VPC routing:** If enhanced VPC routing is turned on, your VPC must be able to reach the Identity Center services. Interface VPC endpoints are the recommended way to provide this connectivity. Set up this connectivity before users sign in. For more information about enhanced VPC routing requirements for Identity Center, see Using AWS IAM Identity Center authentication with enhanced VPC routing.

How Identity Center authentication works

Identity Center authentication for Amazon Redshift uses the following workflow:

1. Your application calls the `GetIdentityCenterAuthToken` API using identity-enhanced credentials that contain embedded user identity information.
2. Amazon Redshift validates the Identity Center identity and generates an encrypted authorized token scoped to specific clusters or workgroups. See example IAM policies.
3. Your application uses this token to connect to the specified Amazon Redshift cluster or workgroup.
4. The Amazon Redshift data plane validates the token and grants access based on the Identity Center user's permissions within the Identity Center application.

> **Important**
>
> This API requires identity-enhanced credentials. For more information, see Using identity-enhanced IAM role sessions.
> If you call the API without identity-enhanced credentials, you will receive an `UnsupportedOperationFault` error.

GetIdentityCenterAuthToken API operations

Amazon Redshift provides two separate `GetIdentityCenterAuthToken` API operations: one for provisioned clusters and one for serverless workgroups. Both operations have the same name but accept different parameters depending on the target resource type.

GetIdentityCenterAuthToken for provisioned clusters

For provisioned Amazon Redshift clusters, use the `GetIdentityCenterAuthToken` API in the Amazon Redshift service to generate authorized token.

Request syntax

```
{
   "ClusterIds": [ "string" ]
}
```

Request parameters

ClusterIds

A list of Amazon Redshift cluster identifiers that the token will be authorized to access. The token can only be used to authenticate with the clusters specified in this list.

Type: Array of strings

Length constraints: Minimum of 1 item. Maximum of 20 items.

Required: Yes

CLI examples

Example: Get authorized token for a single cluster

```
aws redshift get-identity-center-auth-token \
    --cluster-ids my-redshift-cluster
```

Example: Get authorized token for multiple clusters

```
aws redshift get-identity-center-auth-token \
    --cluster-ids my-cluster-1 my-cluster-2
```

GetIdentityCenterAuthToken for serverless workgroups

For Amazon Redshift Serverless workgroups, use the `GetIdentityCenterAuthToken` API in the Amazon Redshift Serverless service to generate authorized token.

Request syntax

```
{
   "WorkgroupNames": [ "string" ]
}
```

Request parameters

WorkgroupNames

A list of Amazon Redshift Serverless workgroup names that the token will be authorized to access. The token can only be used to authenticate with the workgroups specified in this list.

Type: Array of strings

Length constraints: Minimum of 1 item. Maximum of 20 items.

Required: Yes

CLI examples

Example: Get authorized token for a single workgroup

```
aws redshift-serverless get-identity-center-auth-token \
    --workgroup-names my-workgroup
```

Example: Get authorized token for multiple workgroups

```
aws redshift-serverless get-identity-center-auth-token \
```

```
    --workgroup-names workgroup-1 workgroup-2
```

Response syntax

Both APIs return the same response structure:

```
{
   "AuthorizedToken": "string",
   "ExpirationTime": "timestamp"
}
```

Response parameters

AuthorizedToken

An encrypted authorized token that contains the user identity information and the list of authorized clusters or workgroups. This token should be treated as sensitive data.

Type: String

ExpirationTime

The date and time when the token expires, in UTC. Tokens are valid for 1 hour from the time of generation.

Type: Timestamp

Example response

```
{
    "AuthorizedToken":
 "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ1c2VyQGV4YW1wbGUuY29tIiwiaWF0IjoxNjQwOTk1MjAw
    "ExpirationTime": "2024-01-01T12:00:00Z"
}
```

Driver integration

Amazon Redshift drivers support Identity Center authentication through direct token usage:

Direct token usage

After calling the `GetIdentityCenterAuthToken` API to obtain a token, use the `IdpTokenAuthPlugin` with the `SUBJECT_TOKEN` token type.

Connection Configuration:

```
plugin_name = com.amazon.redshift.plugin.IdpTokenAuthPlugin
token_type = SUBJECT_TOKEN
token = {encrypted_token_from_api_response}
```

For detailed information about Identity Center authentication plugins and driver configuration, see Connecting to an Amazon Redshift cluster.

Java code example

Sample Java code to connect using Identity Center authentication:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

// Get token from GetIdentityCenterAuthToken API
String token = "your_encrypted_token_from_api_response";

// Configure connection properties
Properties props = new Properties();
props.setProperty("user", "your_username");
props.setProperty("plugin_name", "com.amazon.redshift.plugin.IdpTokenAuthPlugin");
props.setProperty("token_type", "SUBJECT_TOKEN");
props.setProperty("token", token);

// Connect to Redshift
String url = "jdbc:redshift://your-cluster.region.redshift.amazonaws.com:5439/
your_database";
try (Connection conn = DriverManager.getConnection(url, props)) {
    // Use connection
    System.out.println("Connected successfully!");
```

```
} catch (SQLException e) {
    e.printStackTrace();
}
```

IAM policy requirements

To use Identity Center authentication with Amazon Redshift, specific IAM permissions are required beyond the standard permissions needed for connecting to Amazon Redshift clusters and workgroups.

API permissions

For provisioned clusters, your enhanced IAM role session must have:

- `redshift:GetIdentityCenterAuthToken` on cluster ARNs (format: `arn:aws:redshift:region:account:cluster:cluster-name`)

For serverless workgroups, your enhanced IAM role session must have:

- `redshift-serverless:GetIdentityCenterAuthToken` on workgroup ARNs (format: `arn:aws:redshift-serverless:region:account:workgroup/workgroup-name`)

Example IAM policies

Example policy for provisioned clusters:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "redshift:GetIdentityCenterAuthToken"
            ],
            "Resource": [
                "arn:aws:redshift:us-east-1:123456789012:cluster:my-cluster"
            ]
        }
```

```
    ]
}
```

Example policy for serverless workgroups:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "redshift-serverless:GetIdentityCenterAuthToken"
            ],
            "Resource": [
                "arn:aws:redshift-serverless:us-east-1:123456789012:workgroup/my-
workgroup"
            ]
        }
    ]
}
```

Example policy for multiple resources:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "redshift:GetIdentityCenterAuthToken"
            ],
            "Resource": [
                "arn:aws:redshift:*:123456789012:cluster/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "redshift-serverless:GetIdentityCenterAuthToken"
            ],
            "Resource": [
                "arn:aws:redshift-serverless:*:123456789012:workgroup/*"
            ]
```

```
        }
    ]
}
```

Regional availability

Identity Center authentication is available in the following AWS regions:

- Commercial regions: All supported Amazon Redshift regions
- AWS GovCloud: Available in us-gov-east-1 and us-gov-west-1
- China regions: Available in cn-north-1 and cn-northwest-1

> **Note**
>
> Feature availability may vary during initial rollout.

Troubleshooting Identity Center authentication

Use the following sections to help troubleshoot problems that you have with Identity Center authentication.

Topics

- API errors
- Connection errors

API errors

The `GetIdentityCenterAuthToken` API can return the following errors:

- **UnsupportedOperationFault** – The request was made without identity-enhanced credentials or Identity Center identity propagation is not supported in the current configuration.
- **ClusterNotFoundFault (redshift:GetIdentityCenterAuthToken)** – One or more of the specified cluster identifiers do not exist or you do not have permission to access them.
- **WorkgroupNotFoundFault (redshift-serverless:GetIdentityCenterAuthToken)** – One or more of the specified workgroup names do not exist or you do not have permission to access them.

- **InvalidParameterValueException** – The request parameters are invalid, such as an empty list of clusters or workgroups, or more than 20 items in the list.

Connection errors

When connecting to Amazon Redshift using Identity Center tokens, you may encounter the following errors:

- **Token credentials expired or invalid** – Generate a new token using the `GetIdentityCenterAuthToken` API and update your connection configuration.
- **invalid token** – Verify the token format and ensure it was generated for the correct cluster or workgroup.
- **TYPE_INVALID_TOKEN** – Ensure you're using `token_type = SUBJECT_TOKEN` in your connection parameters.
- **identity provider type "awsidc" is not supported** – Verify your Amazon Redshift cluster or workgroup supports Identity Center authentication and update to a compatible version.
- **AWS IdC identity provider does not exist** – Configure Identity Center integration for your Amazon Redshift resource through the AWS console.
- **Invalid scope. User's credentials are not authorized to connect to Amazon Redshift** – Verify the user has appropriate permissions in Identity Center and the token was generated for the correct resources.
- **Connection failures with enhanced VPC routing** – When enhanced VPC routing is turned on, Amazon Redshift reaches the Identity Center services through your VPC. Verify that your VPC has interface VPC endpoints for those services. Confirm that private DNS names are turned on for each endpoint. Confirm that the endpoint security groups allow inbound traffic on TCP port 443. For more information about enhanced VPC routing requirements for Identity Center, see Using AWS IAM Identity Center authentication with enhanced VPC routing.

Networking tasks

You can perform networking tasks like customizing your connection to a Redshift database. You might want to do this to control traffic for security or other purposes. You can also perform DNS-related tasks, like setting up a custom domain name for your Redshift resources. These configuration tasks are available to you if you have an Amazon Redshift provisioned cluster or with an Amazon Redshift Serverless workgroup.

Topics

- Custom domain names for client connections
- Redshift-managed VPC endpoints
- Redshift resources in a VPC
- Controlling network traffic with Redshift enhanced VPC routing

Custom domain names for client connections

You can create a custom domain name, also known as a custom URL, for both your Amazon Redshift cluster and Amazon Redshift Serverless workgroup. It's an easy-to-read DNS record that routes SQL client connections to your endpoint. You can configure it for an existing cluster or workgroup at any time. It provides several benefits:

- The custom domain name is a more simple string than the default URL, which typically includes the cluster name or the workgroup name and the region. It's easier to recall and use.
- You can quickly route traffic to a new cluster or workgroup in a fail-over case, for example. This makes it so clients don't have to make a configuration change when they reconnect. Connections can be re-routed centrally, with minimal disruption.
- You can avoid sharing private information like a server name in a connection URL. You can hide it in a custom URL.

When you set up a custom domain name using a CNAME, there isn't any additional charge from Amazon Redshift. You may be billed from your DNS provider for a domain name, if you create a new one, but this cost is typically small.

Registering a domain name

Setting up the custom domain name consists of a several tasks: These include registering the domain name with your DNS provider and creating a certificate. After you perform these pieces of work, you configure the custom domain name in the Amazon Redshift console, or in the Amazon Redshift Serverless console, or configure it with AWS CLI commands.

You must have a registered internet domain name to configure a custom domain name in Amazon Redshift. You can register an internet domain using Route 53, or using a third-party domain registration provider. You complete these tasks outside of the Amazon Redshift console. A registered domain is a prerequisite for completing the remaining procedures to create a custom domain.

> **Note**
>
> If you're using a provisioned cluster, prior to performing the steps to configure the custom domain name, it must be relocation enabled. For more information, see Relocating a cluster. This step isn't required for Amazon Redshift Serverless.

The custom domain name typically includes the root domain and a subdomain, like `mycluster.example.com`. To configure it, perform the following steps:

Create a DNS CNAME entry for your custom domain name

1. Register a root domain, for example `example.com`. Optionally, you can use an existing domain. Your custom name can be limited by restrictions on particular characters, or other naming validation. For more information about registering a domain with Route 53, see Registering a new domain.
2. Add a DNS CNAME record that points your custom domain name to the Redshift endpoint for your cluster or workgroup. You can find the endpoint in the properties for the cluster or workgroup, in the Redshift console or in the Amazon Redshift Serverless console. Copy the **JDBC URL** that's available in the cluster or workgroup properties, under **General information**. The URLs appear like the following:
   - For an Amazon Redshift cluster: `redshift-cluster-sample.abc123456.us-east-1.redshift.amazonaws.com`

- For an Amazon Redshift Serverless workgroup: `endpoint-name.012345678901.us-east-1-dev.redshift-serverless-dev.amazonaws.com`

If the URL has a JDBC prefix, remove it.

> **Note**
>
> DNS records are subject to availability, because each name must be unique and available for use within your organization.

Limitations

There are a couple restraints regarding creating CNAME records for a custom domain:

- Creating multiple custom domain names for the same provisioned cluster or Amazon Redshift Serverless workgroup isn't supported. You can associate only one CNAME record.
- Associating a CNAME record with more than one cluster or workgroup isn't supported. The CNAME for each Redshift resource must be unique.

After you register your domain and create the CNAME record, you select a new or existing certificate. You perform this step using AWS Certificate Manager:

We recommend that you create a DNS validated certificate that meets eligibility for managed renewal, which is available with AWS Certificate Manager. Managed renewal means that ACM either renews your certificates automatically or it sends you email notices when expiration is approaching. For more information, see Managed renewal for ACM certificates.

Requesting a certificate for a domain name

Amazon Redshift or Amazon Redshift Serverless require a validated Secure Sockets Layer (SSL) certificate for a custom endpoint to keep communication secure and to verify ownership of the domain name. You can use the your AWS Certificate Manager account with an AWS KMS key for secure certificate management. Security validation includes full host-name verification (*sslmode=verify-full*).

Certificate renewals are managed by Amazon Redshift only when you choose DNS validation, rather than email validation. If you use email validation, you can use the certificate, but you must

perform renewal yourself, prior to its expiration. We recommend that you choose DNS validation for your certificate. You can monitor expiration dates of imported certificates in AWS Certificate Manager.

Request a certificate from ACM for a domain name

1. Sign in to the AWS Management Console and open the ACM console at https://console.aws.amazon.com/acm/.
2. Choose **Request a certificate**.
3. Enter your custom domain name in the **Domain name** field.

   > **Note**
   >
   > You can specify many prefixes, in addition to the certificate domain, in order to use a single certificate for multiple custom-domain records. To illustrate, you can use additional records like `one.example.com`, `two.example.com`, or a wildcard DNS record like `*.example.com` with the same certificate.

4. Choose **Review and request**.
5. Choose **Confirm and request**.
6. For a valid request, a registered owner of the internet domain must consent to the request before ACM issues the certificate. Make sure the status appears as **Issued** in the ACM console, when you're finished with the steps.

Configuring a custom domain

You can use the Amazon Redshift or Amazon Redshift Serverless console to create your custom domain URL. If you haven't configured it, the **Custom domain name** property appears as a dash (**–**) under **General information**. After you create your CNAME record and the certificate, you associate the custom domain name for the cluster or workgroup.

In order to create a custom domain association, the following IAM permissions are required:

- `redshift:CreateCustomDomainAssociation` – You can restrict permission to a specific cluster by adding its ARN.
- `redshiftServerless:CreateCustomDomainAssociation` – You can restrict permission to a specific workgroup by adding its ARN.

- `acm:DescribeCertificate`

As a best practice, we recommend attaching permissions policies to an IAM role and then assigning it to users and groups as needed. For more information, see Identity and access management in Amazon Redshift.

You assign the custom domain name by performing the following steps.

1. Choose the cluster in the Redshift console, or the workgroup in the Amazon Redshift Serverless console, and choose **Create custom domain name** under the **Action** menu. A dialogue appears.
2. Enter the custom domain name.
3. Select the ARN from AWS Certificate Manager for the **ACM Certificate**. Confirm your changes. Per the guidance in the steps you took to create the certificate, we recommend that you choose a DNS validated certificate that's eligible for managed renewal through AWS Certificate Manager.
4. Verify in the cluster properties that the **Custom domain name** and **Custom domain certificate ARN** are populated with your entries. The **Custom domain certificate expiry date** is also listed.

After the custom domain is configured, using `sslmode=verify-full` works only for the new, custom domain. It doesn't work for the default endpoint. But you can can still connect to the default endpoint by using other ssl modes, such as `sslmode=verify-ca`.

Note

As a point of reminder, cluster relocation isn't a prerequisite for configuring additional Redshift networking features. You don't have to turn it on to enable the following:

- **Connecting from a cross-account or cross-region VPC to Redshift** – You can connect from one AWS virtual private cloud (VPC) to another that contains a Redshift database. This makes it easier to manage, for example, client access from disparate accounts or VPCs, without having to provide local VPC access to identities connecting to the database. For more information, see Connecting to Amazon Redshift Serverless from a Redshift VPC endpoint in another account or region.
- **Setting up a custom domain name** – You can create a custom domain name, as described in this topic, to make the endpoint name more relevant and simple.

Connecting to your Amazon Redshift provisioned cluster or Amazon Redshift Serverless workgroup

To connect with a custom domain name, the following IAM permissions are required for a provisioned cluster: `redshift:DescribeCustomDomainAssociations`. For Amazon Redshift Serverless, you don't have to add permissions.

As a best practice, we recommend attaching permissions policies to an IAM role and then assigning it to users and groups as needed. For more information, see Identity and access management in Amazon Redshift.

After you complete the steps to create your CNAME and assign it to your cluster or workgroup in the console, you can provide the custom URL in the connection properties of your SQL client. Note that there can be a delay from DNS propagation immediately following the creation of a CNAME record.

1. Open a SQL client. For example, you can use SQL/Workbench J. Open the properties for a connection, and add the custom domain name for the connection string. For example, `jdbc:redshift://mycluster.example.com:5439/dev?sslmode=verify-full`. In this example, `dev` specifies the default database.
2. Add the **Username** and **Password** for your database user.
3. Test the connection. Your ability to query database resources such as specific tables can vary, based on the permissions granted to the database user or granted to the Amazon Redshift database roles assigned.

   Note that you might have to set your cluster or workgroup to be publicly accessible to connect to it if it's in a VPC. You can change this setting in the network properties.

> **Note**
>
> Connections to a custom domain name are supported with JDBC, ODBC, and Python drivers.

Renaming a cluster that has a custom domain assigned

> **Note**
>
> This series of steps doesn't apply to an Amazon Redshift Serverless workgroup. You can't change the workgroup name.

In order to rename a cluster that has a custom domain name, the `acm:DescribeCertificate` IAM permission is required.

1. Go to the Amazon Redshift console and choose the cluster whose name you want to change. Choose **Edit** to edit the cluster properties.
2. Edit the **Cluster identifier**. You can also change other properties for the cluster. Then choose **Save changes**.
3. After the cluster is renamed, you have to update the DNS record to change the CNAME entry for the custom domain to point to the updated Amazon Redshift endpoint.

Describing custom domain associations

Use the commands in this section to get a list of custom domain names associated with a specific provisioned cluster or with an Amazon Redshift Serverless workgroup.

You need the following permissions:

- For a provisioned cluster: `redshift:DescribeCustomDomainAssociations`
- For an Amazon Redshift Serverless workgroup: `redshiftServerless:ListCnameAssociations`

As a best practice, we recommend attaching permissions policies to an IAM role and then assigning it to users and groups as needed. For more information, see Identity and access management in Amazon Redshift.

The following shows a sample command to list the custom domain names for a given Amazon Redshift cluster:

```
aws redshift describe-custom-domain-associations --custom-domain-name customdomainname
```

You can run this command when you have a custom domain name enabled to determine the custom domain names associated with the cluster. For more information about the CLI command for for describing custom domain associations, see describe-custom-domain-associations.

Similarly, the following shows a sample command to list the custom domain names for a given Amazon Redshift Serverless workgroup. There are a few different ways to do this. You can provide only the custom domain name:

```
aws redshift-serverless list-custom-domain-associations --custom-domain-
name customdomainname
```

You can also get the associations by providing only the certificate ARN:

```
aws redshift-serverless list-custom-domain-associations --custom-domain-certificate-
arn certificatearn
```

You can run these commands when you have a custom domain name enabled to determine the custom domain names associated with the workgroup. You can also run a command to get the properties of a custom domain association. To do this, you must provide the custom domain name and workgroup name as parameters. It returns the certificate ARN, the workgroup name, and the custom domain's certificate expiration time:

```
aws redshift-serverless get-custom-domain-association --workgroup-name workgroupname --
custom-domain-name customdomainname
```

For more information about CLI reference commands available for Amazon Redshift Serverless, see redshift-serverless.

Associating a custom domain with a different certificate

In order to change the certificate association for a custom domain name, the following IAM permissions are required:

- `redshift:ModifyCustomDomainAssociation`
- `acm:DescribeCertificate`

As a best practice, we recommend attaching permissions policies to an IAM role and then assigning it to users and groups as needed. For more information, see Identity and access management in Amazon Redshift.

Use the following command to associate the custom domain with a different certificate. The `--custom-domain-name` and `custom-domain-certificate-arn` arguments are mandatory. The ARN for the new certificate must be different than the existing ARN.

```
aws redshift modify-custom-domain-association --cluster-id redshiftcluster --custom-domain-name customdomainname --custom-domain-certificate-arn certificatearn
```

The following sample shows how to associate the custom domain with a different certificate for an Amazon Redshift Serverless workgroup.

```
aws redshift-serverless modify-custom-domain-association --workgroup-name redshiftworkgroup --custom-domain-name customdomainname --custom-domain-certificate-arn certificatearn
```

There is a maximum delay of 30 seconds before you can connect to the cluster. Part of the delay occurs as the Amazon Redshift cluster updates its properties, and there is some additional delay as DNS is updated. For more information about the API and each property setting, see ModifyCustomDomainAssociation.

Deleting a custom domain

To delete the custom domain name, the user must have permissions for the following actions:

- For a provisioned cluster: `redshift:DeleteCustomDomainAssociation`
- For an Amazon Redshift Serverless workgroup: `redshiftServerless:DeleteCustomDomainAssociation`

On the console

You can delete the custom domain name by selecting the **Actions** button and choosing **Delete custom domain name**. After you do this, you can still connect to the server by updating your tools to use the endpoints listed in the console.

Using a CLI command

The following sample shows how to delete the custom domain name. The delete operation requires that you provide the existing custom domain name for the cluster.

```
aws redshift delete-custom-domain-association --cluster-id redshiftcluster --custom-
domain-name customdomainname
```

The following sample shows how to delete the custom domain name for an Amazon Redshift Serverless workgroup. The custom domain name is a required parameter.

```
aws redshift-serverless delete-custom-domain-association --workgroup-name workgroupname
 --custom-domain-name customdomainname
```

For more information, see DeleteCustomDomainAssociation.

Redshift-managed VPC endpoints

By default, an Amazon Redshift cluster or an Amazon Redshift Serverless workgroup is provisioned in a virtual private cloud (VPC). The VPC can be accessed from another VPC or subnet when you either allow public access or set up an internet gateway, a NAT device, or an AWS Direct Connect connection to route traffic to it. You can also access a cluster or workgroup by setting up a Redshift-managed VPC endpoint (powered by AWS PrivateLink).

You can set up a Redshift-managed VPC endpoint as a private connection between a VPC that contains a cluster or workgroup and a VPC where a client tool is running. If the cluster or workgroup is in another account, the account owner (grantor) must grant access to the connecting account (grantee). With this approach, you can access the data warehouse without using a public IP address or routing traffic through the internet.

These are common reasons to allow access using a Redshift-managed VPC endpoint:

- AWS account A wants to allow a VPC in AWS account B to have access to a cluster or workgroup.
- AWS account A wants to allow a VPC that is also in AWS account A to have access to a cluster or workgroup.
- AWS account A wants to allow a different subnet in the VPC within AWS account A to have access to a cluster or workgroup.

The workflow to set up a Redshift-managed VPC endpoint to access a cluster or workgroup in another account is as follows:

1. The owner account grants access authorization to another account and specifies the AWS account ID and VPC identifier (or all VPCs) of the grantee.

2. The grantee account is notified that they have permission to create a Redshift-managed VPC endpoint.
3. The grantee account creates a Redshift-managed VPC endpoint.
4. The grantee account accesses the cluster or workgroup of the owner account using the Redshift-managed VPC endpoint.

You can do this using the Amazon Redshift console, the AWS CLI, or the Amazon Redshift API.

Considerations when using Redshift-managed VPC endpoints

> **Note**
>
> To create or modify Redshift-managed VPC endpoints, you need permission `ec2:CreateVpcEndpoint` or `ec2:ModifyVpcEndpoint` in your IAM policy, in addition to other permissions specified in the AWS managed policy `AmazonRedshiftFullAccess`.

When using Redshift-managed VPC endpoints, keep the following in mind:

- If you're using a provisioned cluster, it must have the RG or RA3 node type. An Amazon Redshift Serverless workgroup works for setting up a VPC endpoint too.
- For provisioned clusters, make sure that the cluster is enabled for either cluster relocation or Multi-AZ. For information about requirements to turn on cluster relocation, see Relocating a cluster. For information about enabling Multi-AZ, see Setting up Multi-AZ when creating a new cluster.
- Make sure that the cluster or workgroup to access through its security group is available within the valid port ranges 5431-5455 and 8191-8215. The default is 5439.
- You can modify the VPC security groups associated with an existing Redshift-managed VPC endpoint. To modify other settings, delete the current Redshift-managed VPC endpoint and create a new one.
- The number of Redshift-managed VPC endpoints that you can create is limited to your VPC endpoint quota.
- The Redshift-managed VPC endpoints aren't accessible from the internet. A Redshift-managed VPC endpoint is accessible only within the VPC where the endpoint is provisioned or from any VPCs peered with the VPC where the endpoint is provisioned as permitted by the route tables and security groups.

- You can't use the Amazon VPC console to manage Redshift-managed VPC endpoints.
- When you create a Redshift-managed VPC endpoint for a provisioned cluster, the VPC you choose must have a subnet group. To create a subnet group, see Creating a cluster subnet group.
- If an Availability Zone is down, Amazon Redshift does not create a new elastic network interface in another Availability Zone. You might need to create a new endpoint in this case.

For information about quotas and naming constraints, see Quotas and limits in Amazon Redshift.

For information about pricing, see AWS PrivateLink pricing.

Granting access to a VPC

If the VPC that you want to access your cluster or workgroup is in another AWS account, make sure to authorize it from the owner's (grantor's) account.

To allow a VPC in another AWS account to have access to your cluster or workgroup

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Clusters**. For Amazon Redshift Serverless, choose **Serverless dashboard**.
3. For a cluster that you want to allow access to, view the details by choosing the cluster name. Choose the **Properties** tab of the cluster.

   The **Granted accounts** section displays the accounts and corresponding VPCs that have access to your cluster. For an Amazon Redshift Serverless workgroup, choose the workgroup. **Granted accounts** are available under the **Data access** tab.
4. Choose **Grant access** to display a form to enter **Grantee information** to add an account.
5. For **AWS account ID**, enter the ID of the account you are granting access. You can grant access to specific VPCs or all VPCs in the specified account.
6. Choose **Grant access** to grant access.

Creating a Redshift-managed VPC endpoint

If you own a cluster or workgroup, or you have been granted access to manage it, you can create a Redshift-managed VPC endpoint for it.

To create a Redshift-managed VPC endpoint

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Configurations**.

   The **Configurations** page displays the Redshift-managed VPC endpoints that have been created. To view details for an endpoint, choose its name. For Amazon Redshift Serverless, the VPC endpoints are under the **Data access** tab, when you choose the workgroup.
3. Choose **Create endpoint** to display a form to enter information about the endpoint to add.
4. Enter values for **Endpoint name**, the 12-digit **AWS account ID**, the **Virtual private cloud (VPC)** where the endpoint is located, the **Subnet** and the **VPC security group**.

   The subnet in **Subnet** defines the subnets and IP addresses where Amazon Redshift deploys the endpoint. Amazon Redshift chooses a subnet that has IP addresses available for the network interface associated with the endpoint.

   The security group rules in **VPC security group** define the ports, protocols, and sources for inbound traffic that you are authorizing for your endpoint. You allow access to the selected port via the security group or the CIDR range where your workloads run.
5. Choose **Create endpoint** to create the endpoint.

After your endpoint is created, you can access the cluster or workgroup through the URL shown in **Endpoint** URL in the configuration settings for your Redshift-managed VPC endpoint.

Redshift resources in a VPC

You can launch an Amazon Redshift cluster or an Amazon Redshift Serverless workgroup in a VPC on the EC2-VPC platform based on the Amazon VPC service. For more information, see Use EC2 to create your cluster.

> **Note**
>
> Launching clusters and Serverless workgroups into dedicated tenancy VPCs isn't supported. For more information, see Dedicated instances in the *Amazon VPC User Guide*.

When provisioning resources in a VPC, you must do the following:

- **Provide VPC information.**

  When you create a provisioned cluster in your VPC, you must provide your VPC information by creating a cluster subnet group. This information includes the VPC ID and a list of subnets in your VPC. When you launch a cluster, you provide the subnet group so that Redshift can provision it in one of the subnets in the VPC. With Amazon Redshift Serverless, the process is similar. You assign subnets directly to your Serverless workgroup. But in the case of Serverless you don't create a subnet group. For more information about creating subnet groups in Amazon Redshift, see Subnets for Redshift resources. For more information about setting up the VPC, see Getting started with Amazon VPC in the *Amazon VPC Getting Started Guide*.

- **Optionally, configure the accessibility options.**

  Provisioned clusters and serverless workgroups in Amazon Redshift are private by default. If you configure your provisioned cluster or serverless workgroup to be publicly accessible, Amazon Redshift uses an elastic IP address for the external IP address. An elastic IP address is a static IP address. With it, you can change your underlying configuration without affecting the IP address that clients use to connect. This approach can be helpful for situations such as recovery after a failure. Whether you create an elastic IP address depends on your availability zone relocation setting. There are two options:

  1. If you have availability zone relocation turned on and you want to enable public access, you don’t specify an elastic IP address. An elastic IP address managed by Amazon Redshift is assigned. It's associated with your AWS account.
  2. If you have availability zone relocation turned off and you want to enable public access, you can opt to create an elastic IP address for the VPC in Amazon EC2, prior to launching your Amazon Redshift cluster or workgroup. If you don't create an IP address, Amazon Redshift provides a configured elastic IP address to use for the VPC. This elastic IP address is managed by Amazon Redshift and isn't associated with your AWS account.

  For more information, see Elastic IP addresses in the *Amazon EC2 User Guide*.

  In some cases, you might have a publicly accessible cluster in a VPC and you want to connect to it by using the private IP address from within the VPC. If so, set the following VPC parameters to `true`:

  - `DNS resolution`
  - `DNS hostnames`

Note that with Amazon Redshift Serverless, you can't connect in this manner.

Suppose that you have a publicly accessible provisioned cluster in a VPC but don't set those parameters to `true` in the VPC. In these cases, connections made from within the VPC resolve to the elastic IP address of the resource instead of the private IP address. We recommend that you set these parameters to `true` and use the private IP address for a publicly accessible cluster when connecting from within the VPC. For more information, see Using DNS with your VPC in the *Amazon VPC User Guide.*

> **Note**
>
> If you have an existing publicly accessible cluster in a VPC, connections from within the VPC continue to use the elastic IP address to connect to it, until you resize it, if it's a provisioned cluster. This occurs even with the preceding parameters set. Any new clusters created follow the new behavior of using the private IP address when connecting to a publicly accessible cluster from within the same VPC.

The elastic IP address is an external IP address for accessing a resource outside of a VPC. For a provisioned cluster, it isn't related to the **Public IP addresses** and **Private IP addresses** that are displayed in the Amazon Redshift console under **Node IP addresses**. The public and private cluster node IP addresses appear regardless of whether a cluster is publicly accessible or not. They're used only in certain circumstances to configure ingress rules on the remote host. These circumstances occur when you load data from an Amazon EC2 instance or other remote host using a Secure Shell (SSH) connection. For more information, see Step 1: Retrieve the cluster public key and cluster node IP addresses in the *Amazon Redshift Database Developer Guide.*

> **Note**
>
> Node IP addresses don't apply for a Redshift Serverless workgroup.

The option to associate a provisioned cluster with an elastic IP address is available when you create the cluster or restore the cluster from a snapshot. In some cases, you might want to associate the cluster with an elastic IP address or change an elastic IP address that is associated with the cluster. To attach an elastic IP address after the cluster is created, first update the

cluster so that it is not publicly accessible, then make it both publicly accessible and add an Elastic IP address in the same operation.

For more information about how to make a provisioned cluster or Amazon Redshift Serverless workgroup publicly accessible, and have an Elastic IP address assigned, see Public accessibility with default or custom security group configuration.

- **Associate a VPC security group.**

  You grant inbound access using a VPC security group. For more information, see Configuring security group communication settings for Amazon Redshift clusters, which provides guidance on configuring inbound and outbound rules between a client and a provisioned cluster or an Amazon Redshift Serverless workgroup. Another resource that helps you understand security groups is Security in your VPC in the *Amazon VPC User Guide*

Restoring a snapshot of a provisioned cluster or Serverless workgroup in a VPC

A snapshot of a cluster or Serverless workgroup in a VPC can only be restored in a VPC, not outside the VPC. You can restore it in the same VPC or another VPC in your account. For more information about snapshots, see Amazon Redshift snapshots and backups.

Creating a Redshift provisioned cluster or Amazon Redshift Serverless workgroup in a VPC

The following are the general steps how you can deploy a cluster or workgroup in your virtual private cloud (VPC).

To create a cluster or Serverless workgroup in a VPC

1. Configure a VPC – You can create your Redshift resources either in the default VPC for your account, if your account has one, or in a VPC that you created. For more information, see Use EC2 to create your cluster. To create a VPC, see Subnets for your VPC in the *Amazon VPC User Guide.* Make a note of the VPC identifier, subnet, and subnet's Availability Zone. You need this information when you launch your cluster or workgroup.

> **Note**
>
> You must have at least one subnet defined in your VPC, so you can add it to the subnet group in the next step. For more information about adding a subnet to your VPC, see Adding a subnet to your VPC in the *Amazon VPC User Guide.*

2. Create an Amazon Redshift cluster subnet group to specify which subnet your Amazon Redshift cluster can use in the VPC. For Redshift Serverless, you don't create a subnet group, but rather assign a collection of subnets to your workgroup when you create it. You can perform this in the **Serverless dashboard** when you create a workgroup.

   You can create a subnet group using either the Amazon Redshift console or programmatically. For more information, see Subnets for Redshift resources.
3. Authorize access for inbound connections in a VPC security group that you associate with the cluster or workgroup. You can enable a client outside the VPC (on the public internet) to connect to the cluster. To do this, you associate the cluster with a VPC security group that grants inbound access. For more information, see Configuring security group communication settings for an Amazon Redshift cluster or an Amazon Redshift Serverless workgroup.
4. Follow the steps to create a cluster in the Redshift provisioned console or a workgroup or in the Amazon Redshift Serverless console. In **Network and security**, specify the **Virtual private cloud (VPC)**, **Cluster subnet group**, and **VPC security group** that you set up.

   For a walkthrough that shows more detailed steps for creating a provisioned data-warehouse cluster, see Get started with Amazon Redshift provisioned data warehouses in the *Amazon Redshift Getting Started Guide*. For more information about creating an Amazon Redshift Serverless workgroup, see Get started with Amazon Redshift Serverless data warehouses in the *Amazon Redshift Getting Started Guide*.

You can follow the Getting Started steps to test the cluster or workgroup by uploading sample data and trying example queries. For more information, see Get started with Amazon Redshift Serverless data warehouses in the *Amazon Redshift Getting Started Guide*.

VPC security groups

When you provision an Amazon Redshift cluster or Amazon Redshift Serverless workgroup, access is restricted by default so nobody has access to it. To grant other users inbound access,

you associate it with a security group. If you are on the EC2-VPC platform, you can either use an existing Amazon VPC security group or define a new one. You then associate it with a cluster or workgroup as described following. If you are on the EC2-Classic platform, you define a security group and associate it with your cluster or workgroup. For more information on using security groups on the EC2-Classic platform, see Amazon Redshift security groups.

A VPC security group consists of a set of rules that control access to an instance on the VPC, such as your cluster. Individual rules set access based either on ranges of IP addresses or on other VPC security groups. When you associate a VPC security group with a cluster or workgroup, the rules that are defined in the VPC security group control access.

Each cluster you provision on the EC2-VPC platform has one or more Amazon VPC security groups associated with it. Amazon VPC provides a VPC security group called default, which is created automatically when you create the VPC. Each cluster that you launch in the VPC is automatically associated with the default VPC security group if you don't specify a different VPC security group when your Redshift resources. You can associate a VPC security group with a cluster when you create the cluster, or you can associate a VPC security group later by modifying the cluster.

The following screenshot shows the default rules for the default VPC security group.

Inbound			
Source	**Protocol**	**Port Range**	**Comments**
The security group ID (sg-*xxxxxxxx*)	All	All	Allow inbound traffic from instances assigned to the same security group
Outbound			
Destination	**Protocol**	**Port Range**	**Comments**
0.0.0.0/0	All	All	Allow all outbound traffic

You can change the rules for the default VPC security group as needed.

If the default VPC security group is enough for you, you don't need to create more. However, you can optionally create additional VPC security groups to better manage inbound access. For example, suppose that you are running a service on an Amazon Redshift cluster or Serverless workgroup, and you have several different service levels you provide to your customers. If you don't want to provide the same access at all service levels, you might want to create separate VPC security groups, one for each service level. You can then associate these VPC security groups with your cluster or workgroups.

You can create up to 100 VPC security groups for a VPC and associate a VPC security group with multiple clusters and workgroups. However, note that there are limits to the number of VPC security groups you can associate with a cluster or workgroup.

Amazon Redshift applies changes to a VPC security group immediately. So if you have associated the VPC security group with a cluster, inbound cluster access rules in the updated VPC security group apply immediately.

You can create and modify VPC security groups at https://console.aws.amazon.com/vpc/. You can also manage VPC security groups programmatically by using the AWS CLI, the Amazon EC2 CLI, and the AWS Tools for Windows PowerShell. For more information about working with VPC security groups, see Security groups for your VPC in the *Amazon VPC User Guide*.

Configuring security group communication settings for an Amazon Redshift cluster or an Amazon Redshift Serverless workgroup

This topic helps you configure your security groups to route and receive network traffic appropriately. The following are a couple common use cases:

- You turn on public accessibility for an Amazon Redshift cluster or an Amazon Redshift Serverless workgroup, but it isn't receiving traffic. For this you must configure an inbound rule to allow traffic to reach it from the internet.
- Your cluster or workgroup isn't publicly accessible, and you use Redshift's pre-configured default VPC security group to allow inbound traffic. But you have a requirement to use a security group other than the default, and this custom security group doesn't allow inbound traffic. You must configure it to allow communication.

The following sections help you choose the correct response for each use case and show you how to configure network traffic per your requirements. You can optionally use the steps to set up communication from other private security groups.

> **Note**
>
> Network traffic settings in most cases aren't configured automatically in Amazon Redshift. This is because they can vary at a granular level, depending on whether the source of traffic is the internet or a private security group, and because security requirements vary.

Public accessibility with default or custom security group configuration

If you are creating or you already have a cluster or workgroup, perform the following configuration steps to make it publicly accessible. This applies both to when you choose the default security group or a custom security group:

1. Find the network settings:
   - For a provisioned Amazon Redshift cluster, choose the **Properties** tab, and then under **Network and security settings**, select the VPC for your cluster.
   - For an Amazon Redshift Serverless workgroup, choose **Workgroup configuration**. Choose the workgroup from the list. Then, under **Data access**, in the **Network and security** panel, choose **edit**.
2. Configure the Internet gateway and route table for your VPC. You start the configuration by choosing the VPC by name. It opens the VPC dashboard. To connect to a publicly accessible cluster or workgroup from the internet, an internet gateway must be attached to the route table. You can configure this by choosing **Route tables** in the VPC dashboard. Confirm that the internet gateway's target is set with source 0.0.0.0/0 or a public IP CIDR. The route table must be associated with the VPC where your cluster resides. For more information regarding setting up internet access for a VPC, like what is described here, see Enable internet access in the Amazon VPC documentation. For more information about configuring a route table, see Configure route tables.
3. After you configure the internet gateway and route table, return to the network settings for Redshift. Open inbound access by choosing the security group and then choosing the **Inbound rules**. Choose **Edit inbound rules**.
4. Choose the **Protocol** and **Port** for the inbound rule, or rules, per your requirements, to allow traffic from clients. For an RG or RA3 cluster, select a port within the ranges 5431-5455 or 8191-8215. When you are finished, save each rule.
5. Edit the **Publicly accessible** setting to enable it. You can do this from your cluster or workgroup's **Actions** menu.

When you turn on the publicly accessible setting, Redshift creates an Elastic IP address. It's a static IP address that's associated with your AWS account. Clients outside the VPC can use it to connect.

For more information about configuring your security group, see Amazon Redshift security groups.

You can test your rules by connecting with a client, perform the following if you're connecting to Amazon Redshift Serverless. After you finish network configuration, connect with your client tool, such as Amazon Redshift RSQL. Using your Amazon Redshift Serverless domain as the host, enter the following:

```
rsql -h workgroup-name.account-id.region.amazonaws.com -U admin -d dev -p 5439
```

Private accessibility with default or custom security group configuration

When you don't communicate through the internet to your cluster or workgroup, it's referred to as *privately* accessible. If you chose the default security group when you created it, the security group includes the following default communication rules:

- An inbound rule that allows traffic from all resources assigned to the security group.
- An outbound rule that allows all outbound traffic. The destination for this rule is 0.0.0.0/0. In classless inter-domain routing (CIDR) notation, it represents all possible IP addresses.

You can view the rules in the console by selecting the security group for your cluster or workgroup.

If your cluster or workgroup and client both use the default security group, there isn't any additional configuration necessary to allow network traffic. But if you delete or change any rules in the default security group for Redshift or the client, this no longer applies. In this case, you must configure rules to allow inbound and outbound communication. A common security-group configuration is the following:

- For a client Amazon EC2 instance:
  - An inbound rule that allows the IP address of the client.
  - An outbound rule that allows the IP address range (CIDR block) of all subnets provided for Redshift usage. Or your can specify 0.0.0.0/0, which is all IP address ranges.
- For your Redshift cluster or workgroup:
  - An inbound rule that allows the client security group.
  - An outbound rule that allows traffic to 0.0.0.0/0. Typically, the outbound rule allows all outbound traffic. Optionally, you can add an outbound rule to allow traffic to the client security group. In this optional case, an outbound rule isn't always required, because response traffic for each request is allowed to reach the instance. For more details regarding request and response behavior, see Security groups in the *Amazon VPC user guide*.

If you change configuration for any subnets or security groups specified for Redshift usage, you might need to change traffic rules accordingly to keep communication open. For more information about creating inbound and outbound rules, see VPC CIDR blocks in the *Amazon VPC user guide*. For more information about connecting to Amazon Redshift from a client, see Configuring connections in Amazon Redshift.

Subnets for Redshift resources

You create a subnet group if you are creating a provisioned cluster in a virtual private cloud (VPC). Each VPC can have one or more subnets, which are subsets of IP addresses within the VPC that enable you to group resources based on your security and operation needs. You create a subnet group to specify a set of subnets in your VPC when you create a provisioned cluster. In the **Provisioned clusters dashboard**, you can find and edit cluster subnet groups under **Configurations**. During initial configuration for a provisioned cluster, you specify the subnet group and Amazon Redshift creates the cluster in one of its subnets. For more information about the VPC service, see the Amazon VPC product detail page.

Subnet configuration for an Amazon Redshift Serverless workgroup is similar to a provisioned cluster, but the steps differ slightly. When you create and set up a Serverless workgroup, you specify subnets for the workgroup, and they're added to a list. You can view the subnets for an existing workgroup by selecting the workgroup properties, in the **Serverless dashboard**. They're available in the **Network and security** properties. For more information, see Creating a workgroup with a namespace.

For more information about creating a VPC, go to Amazon VPC User Guide documentation.

After creating a subnet group for a provisioned cluster, or choosing subnets for a Serverless workgroup, it's possible to remove subnets previously added or to add more. You can make these changes using the console, or using API operations. For more information regarding API operations for a provisioned cluster, see ModifyClusterSubnetGroup. For API operations for a Serverless workgroup, see UpdateWorkgroup.

You can provision a cluster on one of the subnets in the subnet group. A cluster subnet group enables you to specify a set of subnets in your virtual private cloud (VPC).

> **Warning**
>
> During cluster maintenance operations such as classic resize, pause and resume, Multi-AZ failovers, or other events, your provisioned compute nodes might be moved to another

subnet within your Amazon Redshift cluster subnet group. Note that all subnets in a subnet group must have the same Network ACL inbound and outbound rules and the same route-table routes. This ensures connectivity to and from the Amazon Redshift compute resources, so they can communicate and function optimally after such maintenance events. Avoid adding subnets with varying network ACL or route-table configurations to the same Amazon Redshift cluster subnet group.
For more information about configuring subnets, see Subnets for your VPC in the Amazon VPC user guide. For more information about Redshift Multi-AZ deployments, see Multi-AZ deployment in the Redshift management guide. Resizing a cluster is also covered in the Redshift management guide.

Creating a cluster subnet group

The following procedure walks you through how to create a subnet group for a provisioned cluster. You must have at least one cluster subnet group defined to provision a cluster in a VPC.

To create a cluster subnet group for a provisioned cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Configurations**, then choose **Subnet groups**. The list of subnet groups is displayed.
3. Choose **Create cluster subnet group** to display the create page.
4. Enter information for the subnet group, including which subnets to add.
5. Choose **Create cluster subnet group** to create the group with the subnets that you chose.

Note

For information about how to create an Amazon Redshift Serverless workgroup with a collection of subnets, see Creating a workgroup with a namespace or Create a subnet in the Amazon VPC User Guide.

Modifying a cluster subnet group

After you've created a subnet group, you can modify its information on the Amazon Redshift console.The following procedure walks you through how to modify a subnet group for a provisioned cluster.

To modify a cluster subnet group for a provisioned cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Configurations**, then choose **Subnet groups**. The list of subnet groups is displayed.
3. Choose the subnet group to modify.
4. For **Actions**, choose **Modify** to display the details of the subnet group.
5. Update information for the subnet group.
6. Choose **Save** to modify the group.

To change or remove subnets in some cases requires extra steps. For example, this AWS Knowledge Center article, How do I move my provisioned Amazon Redshift cluster into a different subnet?, describes a use case that covers moving a cluster.

Deleting a cluster subnet group for a provisioned cluster

When you're done using a cluster subnet group, you should clean up by deleting the group. The following procedure walks you through the steps to delete a subnet group for a provisioned cluster.

To delete a cluster subnet group

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Configurations**, then choose **Subnet groups**. The list of subnet groups is displayed.
3. Choose the subnet group to delete, then choose **Delete**.

> **Note**
>
> You can't delete a cluster subnet group that is used by a cluster.

Blocking public access to VPCs and subnets

VPC Block Public Access (BPA) is a centralized security feature that you can use to block resources in VPCs and subnets that you own in an AWS Region from reaching the internet or being reached from the internet through internet gateways and egress-only internet gateways. If you turn this feature on in an AWS account, it will, by default, impact any VPC or subnet that Amazon Redshift uses. This means that Amazon Redshift blocks all operations to the public.

When you have VPC BPA turned on and want to use Amazon Redshift APIs over the public internet, you must add an exclusion for using Amazon EC2 APIs for your VPC or subnet. Exclusions can have either of the following modes:

- **Bidirectional**: All internet traffic to and from the excluded VPCs and subnets is allowed.
- **Egress-only**: Outbound internet traffic from the excluded VPCs and subnets is allowed. Inbound internet traffic to the excluded VPCs and subnets is blocked. This only applies when BPA is set to bidirectional.

VPC BPA exclusions designate an entire VPC or specific subnet within a VPC as public-access capable. Network interfaces within that boundary respect the regular VPC networking controls, such as security groups, route tables, and network ACLs, with regard to whether that interface has a route and access to the public internet. For more information about adding exclusions, see Create and delete exclusions in the *Amazon VPC User Guide*.

Provisioned clusters

A subnet group is a combination of subnets from the same VPC. If a subnet group for a provisioned cluster is in an account with VPC BPA turned on, the following capabilities are blocked:

- Creating a public cluster
- Restoring a public cluster
- Modifying a private cluster to be public
- Adding a subnet with VPC BPA turned on to the subnet group when there's at least one public cluster within the group

Serverless clusters

Redshift Serverless doesn't use subnet groups. Instead, each cluster has its own set of subnets. If a workgroup is in an account with VPC BPA turned on, the following capabilities are blocked:

- Creating a public access workgroup
- Modifying a private workgroup to public
- Adding a subnet with VPC BPA turned on to the workgroup when the workgroup is public

Controlling network traffic with Redshift enhanced VPC routing

When you use Amazon Redshift enhanced VPC routing, Amazon Redshift forces all COPY and UNLOAD traffic between your cluster and your data repositories through your virtual private cloud (VPC) based on the Amazon VPC service. By using enhanced VPC routing, you can use standard VPC features, such as VPC security groups, network access control lists (ACLs), VPC endpoints, VPC endpoint policies, internet gateways, and Domain Name System (DNS) servers, as described in the *Amazon VPC User Guide*. You use these features to control the flow of data between your Amazon Redshift cluster and other resources. When you use enhanced VPC routing to route traffic through your VPC, you can also use VPC flow logs to monitor COPY and UNLOAD traffic.

Amazon Redshift clusters and Amazon Redshift Serverless workgroups both support enhanced VPC routing. You can't use enhanced VPC routing with Redshift Spectrum. For more information, see Querying data lake tables with enhanced VPC routing.

If enhanced VPC routing is not turned on, Amazon Redshift routes traffic through the internet, including traffic to other services within the AWS network.

> **Important**
>
> Because enhanced VPC routing affects the way that Amazon Redshift accesses other resources, COPY and UNLOAD commands might fail unless you configure your VPC correctly. You must specifically create a network path between your cluster's VPC and your data resources, as described following.

When you run a COPY or UNLOAD command on a cluster with enhanced VPC routing turned on, your VPC routes the traffic to the specified resource using the *strictest*, or most specific, network path available.

For example, you can configure the following pathways in your VPC:

- **VPC endpoints** – For traffic to an Amazon S3 bucket in the same AWS Region as your cluster or workgroup, you can create a VPC endpoint to direct traffic directly to the bucket. When you use VPC endpoints, you can attach an endpoint policy to manage access to Amazon S3. For more information about using endpoints with Redshift, see Controlling database traffic with VPC endpoints. If you use Lake Formation, you can find more information about establishing a private connection between your VPC and AWS Lake Formation at AWS Lake Formation and interface VPC endpoints (AWS PrivateLink).

  > **Note**
  >
  > When you use Redshift VPC endpoints with Amazon S3 VPC Gateway endpoints, you must enable enhanced VPC routing in Redshift. For more information, see Gateway endpoints for Amazon S3.

- **NAT gateway** – You can connect to an Amazon S3 bucket in another AWS Region, and you can connect to another service within the AWS network. You can also access a host instance outside the AWS network. To do so, configure a network address translation (NAT) gateway, as described in the *Amazon VPC User Guide.*
- **Internet gateway** – To connect to AWS services outside your VPC, you can attach an internet gateway to your VPC subnet, as described in the *Amazon VPC User Guide.* To use an internet gateway, your cluster or workgroup must be publicly accessible to allow other services to communicate it.

For more information, see VPC Endpoints in the Amazon VPC User Guide.

> **Note**
>
> Enhanced VPC routing also affects the calls that Redshift makes to AWS IAM Identity Center when a user signs in. If your cluster or workgroup uses AWS IAM Identity Center authentication, you must create interface VPC endpoints for those services. Otherwise, sign-in fails. For more information about using enhanced VPC routing with AWS IAM Identity Center, see Using AWS IAM Identity Center authentication with enhanced VPC routing.

There is no additional charge for using enhanced VPC routing. You might incur additional data transfer charges for certain operations. These include such operations as UNLOAD to Amazon S3 in a different AWS Region. COPY from Amazon EMR, or Secure Shell (SSH) with public IP addresses. For more information about pricing, see Amazon EC2 Pricing.

Topics

- Controlling database traffic with VPC endpoints
- Turning on enhanced VPC routing
- Querying data lake tables with enhanced VPC routing

Controlling database traffic with VPC endpoints

You can use a VPC endpoint to create a managed connection between your Amazon Redshift cluster or Serverless workgroup in a VPC and Amazon Simple Storage Service (Amazon S3). When you do, COPY and UNLOAD traffic between your database and your data on Amazon S3 stays in your Amazon VPC. You can attach an endpoint policy to your endpoint to more closely manage access to your data. For example, you can add a policy to your VPC endpoint that permits unloading data only to a specific Amazon S3 bucket in your account.

To use VPC endpoints, create a VPC endpoint for the VPC that your data warehouse is in and then turn on enhanced VPC routing. You can turn on enhanced VPC routing when you create your cluster or workgroup, or you can modify a cluster or workgroup in a VPC to use enhanced VPC routing.

A VPC endpoint uses route tables to control the routing of traffic between a cluster or workgroup in the VPC and Amazon S3. All clusters and workgroups in subnets associated with the specified route tables automatically use that endpoint to access the service.

Your VPC uses the most specific, or most restrictive, route that matches your traffic to determine how to route the traffic. For example, suppose that you have a route in your route table for all internet traffic (0.0.0.0/0) that points to an internet gateway and an Amazon S3 endpoint. In this case, the endpoint route takes precedence for all traffic destined for Amazon S3. This is because the IP address range for the Amazon S3 service is more specific than 0.0.0.0/0. In this example, all other internet traffic goes to your internet gateway, including traffic that's destined for Amazon S3 buckets in other AWS Regions.

For more information about creating endpoints, see Create a VPC endpoint in the *Amazon VPC User Guide.*

You use endpoint policies to control access from your cluster or workgroup to the Amazon S3 buckets that hold your data files. For more specific control, you can optionally attach a custom endpoint policy. For more information, see Control access to services using endpoint policies in the *AWS PrivateLink Guide.*

> **Note**
>
> AWS Database Migration Service (AWS DMS) is a cloud service that makes it possible to migrate relational databases, data warehouses, and other types of data stores. It can connect to any AWS source or target database, including an Amazon Redshift database that's VPC enabled, with some configuration restrictions. Supporting Amazon VPC endpoints makes it easier for AWS DMS to maintain end-to-end network security for replication tasks. For more information on using Redshift with AWS DMS, see Configuring VPC endpoints as AWS DMS source and target endpoints in the *AWS Database Migration Service User Guide*.

There is no additional charge for using endpoints. Standard charges for data transfer and resource usage apply. For more information about pricing, see Amazon EC2 Pricing.

Turning on enhanced VPC routing

You can turn on enhanced VPC routing when you create or modify a cluster, and when you create or modify a Amazon Redshift Serverless workgroup.

To work with enhanced VPC routing, your cluster or Serverless workgroup must meet the following requirements and constraints:

- Your cluster must be in a VPC.

  If you attach an Amazon S3 VPC endpoint, the VPC endpoint is used only for access to Amazon S3 buckets in the same AWS Region. To access buckets in another AWS Region (not using the VPC endpoint) or to access other AWS services, make your cluster or Serverless workgroup publicly accessible or use a network address translation (NAT) gateway. For more information, see Creating a Redshift provisioned cluster or Amazon Redshift Serverless workgroup in a VPC.
- You must enable Domain Name Service (DNS) resolution in your VPC. Alternatively, if you're using your own DNS server, make sure that DNS requests to Amazon S3 are resolved correctly to the IP addresses that are maintained by AWS. For more information, see Using DNS with Your VPC in the *Amazon VPC User Guide.*

- DNS hostnames must be enabled in your VPC. DNS hostnames are enabled by default.
- Your VPC endpoint policies must allow access to any Amazon S3 buckets used with COPY, UNLOAD, or CREATE LIBRARY calls in Amazon Redshift, including access to any manifest files involved. For COPY from remote hosts, your endpoint policies must allow access to each host machine. For more information, see IAM Permissions for COPY, UNLOAD, and CREATE LIBRARY in the *Amazon Redshift Database Developer Guide.*

To turn on enhanced VPC routing for a provisioned cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Provisioned clusters dashboard**, then choose **Create cluster** and enter the **Cluster details** properties.
3. To display the **Additional configurations** section, choose to switch off **Use defaults**.
4. Navigate to the **Network and security** section.
5. To turn on **Enhanced VPC routing**, choose **Turn on** to force cluster traffic through the VPC.
6. Choose **Create cluster** to create the cluster. The cluster might take several minutes to be ready to use.

To turn on enhanced VPC routing for an Amazon Redshift Serverless

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Serverless dashboard**, then choose **Create workgroup** and enter the properties for your workgroup.
3. Navigate to the **Network and security** section.
4. Select **Turn on enhanced VPC routing** to route network traffic through the VPC.
5. Choose **Next** and finish entering your workgroup properties until you **Create** the workgroup.

Querying data lake tables with enhanced VPC routing

When enhanced VPC routing is turned on, Amazon Redshift routes traffic through your VPC. Data lake tables store their data in Amazon S3 and their metadata in the AWS Glue Data Catalog, both of which are outside your VPC. For data lake queries to succeed, your cluster or workgroup must

be able to reach Amazon S3 and AWS Glue. If your data lake tables are managed by AWS Lake Formation, it must also be able to reach Lake Formation. Enhanced VPC routing affects the way that Amazon Redshift accesses these external resources, so queries might fail unless you configure your VPC correctly.

Create the following VPC endpoints in the VPC and subnets where your cluster or workgroup runs, associating them with the appropriate route tables. These endpoints are required for both the integrated data lake query engine and Amazon Redshift Spectrum:

- **Amazon S3 gateway endpoint** – Gives your cluster or workgroup a route through the AWS network to Amazon S3 to read data lake data files. For steps, see Gateway endpoints for Amazon S3 in the *Amazon VPC User Guide*.
- **AWS Glue interface endpoint** – Gives your cluster or workgroup a route to the AWS Glue Data Catalog to resolve data lake schemas and tables. For steps, see Creating an interface endpoint in the *Amazon VPC User Guide*.
- **AWS Lake Formation interface endpoint** – Required only if your data lake tables are managed by AWS Lake Formation. Gives your cluster or workgroup a route to Lake Formation. For steps, see Creating an interface endpoint in the *Amazon VPC User Guide*.

The endpoint configuration is the same regardless of how you query external tables, but the way traffic is routed differs depending on the query engine. The following sections describe the behavior for the integrated data lake query engine on RG provisioned clusters and Amazon Redshift Serverless, and for Redshift Spectrum on RA3 and DC2 provisioned clusters.

Integrated data lake query engine for RG provisioned clusters and Amazon Redshift Serverless

RG provisioned clusters and Amazon Redshift Serverless include an integrated data lake query engine. This engine runs on the cluster's or workgroup's own compute resources, within your VPC. When enhanced VPC routing is turned on and the required VPC endpoints are in place, access to Amazon S3 and AWS Glue originates from your in-VPC compute and flows through those VPC endpoints. This traffic stays within your VPC boundary and on the AWS network. This access is authorized based on the IAM role that is attached to your cluster or workgroup. To further manage this traffic, you can modify the IAM role and the policy attached to the Amazon S3 bucket.

Amazon Redshift Spectrum for RA3 and DC2 provisioned clusters

Redshift Spectrum runs on AWS managed resources that are owned by Amazon Redshift but are outside your VPC. As a result, even when enhanced VPC routing is turned on, the traffic that Redshift Spectrum sends to Amazon S3 does not pass through your VPC, but the cluster still requires the Amazon S3 and AWS Glue VPC endpoints described earlier to run data lake queries.

Traffic between Redshift Spectrum and Amazon S3 is securely routed through the AWS private network, outside of your VPC. In-flight traffic is signed using Amazon Signature Version 4 protocol (SIGv4) and encrypted using HTTPS. This traffic is authorized based on the IAM role that is attached to your Amazon Redshift cluster. To further manage Redshift Spectrum traffic, you can modify your cluster's IAM role and your policy attached to the Amazon S3 bucket.

Permissions policy configuration for data lake queries

Consider the following when querying data lake tables in Amazon S3:

- Amazon S3 bucket access policies and IAM roles
- Permissions for assuming the IAM role
- Logging and auditing Amazon S3 access
- Access to AWS Glue or Amazon Athena

Amazon S3 bucket access policies and IAM roles

You can control access to data in your Amazon S3 buckets by using a bucket policy attached to the bucket and by using an IAM role attached to a provisioned cluster.

Redshift Spectrum on provisioned clusters can't access data stored in Amazon S3 buckets that use a bucket policy that restricts access to only specified VPC endpoints. Instead, use a bucket policy that restricts access to only specific principals, such as a specific AWS account or specific users. The integrated data lake query engine on RG provisioned clusters and Amazon Redshift Serverless can access such buckets when enhanced VPC routing is turned on, because its Amazon S3 access flows through your VPC endpoints.

For the IAM role that is granted access to the bucket, use a trust relationship that allows the role to be assumed only by the Amazon Redshift service principal. When attached to your cluster, the role can be used only in the context of Amazon Redshift and can't be shared outside of the cluster. For more information, see Restricting access to IAM roles. A service control policy (SCP) can also be

used to further restrict the role, see Prevent IAM users and roles from making specified changes, with an exception for a specified admin role in the *AWS Organizations User Guide*.

> **Note**
>
> To use Redshift Spectrum, no IAM policies blocking the use of Amazon S3 presigned URLs can be in place. The presigned URLs generated by Amazon Redshift Spectrum are valid for 1 hour so that Amazon Redshift has enough time to load all the files from the Amazon S3 bucket. A unique presigned URL is generated for each file scanned by Redshift Spectrum. For bucket policies that include an `s3:signatureAge` action, make sure to set the value to at least 3,600,000 milliseconds.

The following example bucket policy permits access to the specified bucket owned by AWS account `123456789012`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "BucketPolicyForSpectrum",
            "Effect": "Allow",
            "Principal": {
                "AWS": ["arn:aws:iam::123456789012:role/redshift"]
            },
            "Action": [
                "s3:GetObject",
                "s3:ListBucketVersions",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

Permissions for assuming the IAM role

The role attached to your cluster should have a trust relationship that permits it to be assumed only by the Amazon Redshift service, as shown following.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

For more information, see IAM Policies for Redshift Spectrum in the *Amazon Redshift Database Developer Guide*.

Logging and auditing Amazon S3 access

One benefit of using Amazon Redshift enhanced VPC routing is that all COPY and UNLOAD traffic is logged in the VPC flow logs. Traffic originating from Redshift Spectrum to Amazon S3 doesn't pass through your VPC, so it isn't logged in the VPC flow logs. When Redshift Spectrum accesses data in Amazon S3, it performs these operations in the context of the AWS account and respective role privileges. You can log and audit Amazon S3 access using server access logging in AWS CloudTrail and Amazon S3.

Ensure that the S3 IP ranges are added to your allow list. To learn more about the required S3 IP ranges, see Network isolation.

AWS CloudTrail Logs

To trace all access to objects in Amazon S3, including Redshift Spectrum access, enable CloudTrail logging for Amazon S3 objects.

You can use CloudTrail to view, search, download, archive, analyze, and respond to account activity across your AWS infrastructure. For more information, see Getting Started with CloudTrail.

By default, CloudTrail tracks only bucket-level actions. To track object-level actions (such as `GetObject`), enable data and management events for each logged bucket.

Amazon S3 Server Access Logging

Server access logging provides detailed records for the requests that are made to a bucket. Access log information can be useful in security and access audits. For more information, see How to Enable Server Access Logging in the *Amazon Simple Storage Service User Guide.*

For more information, see the AWS Security blog post How to Use Bucket Policies and Apply Defense-in-Depth to Help Secure Your Amazon S3 Data.

Access to AWS Glue or Amazon Athena

Both Redshift Spectrum and the integrated data lake query engine access your data catalog in AWS Glue or Athena. Another option is to use a dedicated Hive metastore for your data catalog.

The recommended way to reach the AWS Glue Data Catalog is to create an interface VPC endpoint (AWS PrivateLink) for AWS Glue. When you use a VPC interface endpoint, communication between your VPC and AWS Glue is routed within the AWS network. For more information, see Creating an Interface Endpoint.

Alternatively, to reach the public endpoints for AWS Glue and Athena, configure your VPC with an internet gateway or NAT gateway, and configure your VPC security groups to allow outbound traffic to those public endpoints. You can configure the following pathways in your VPC:

- **Internet gateway** –To connect to AWS services outside your VPC, you can attach an internet gateway to your VPC subnet, as described in the *Amazon VPC User Guide.* To use an internet gateway, a provisioned cluster must have a public IP address to allow other services to communicate with it.
- **NAT gateway** –To connect to an Amazon S3 bucket in another AWS Region or to another service within the AWS network, configure a network address translation (NAT) gateway, as described in the *Amazon VPC User Guide.* Use this configuration also to access a host instance outside the AWS network.

Amazon Redshift events

Amazon Redshift tracks cluster 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.

Amazon Redshift provides notification in advance for some events. These events have an event category of `pending`. For example, we send an advance notification if a hardware update is required for one of the nodes in your cluster. You can subscribe to pending events the same as other Amazon Redshift events. For more information, see Amazon Redshift cluster event notification subscriptions.

You can use the Amazon Redshift Management Console, the Amazon Redshift API, or the AWS SDKs to obtain event information. You can obtain a list of all events, or you can apply filters, such as event duration or start and end date, to obtain events information for a specific period.

You can also obtain events that were generated by a specific source type, such as cluster events or parameter group events. The *Source* column shows the resource name and resource type that triggers a given action.

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.

For a list of Amazon Redshift events by source type and category, see the section called "Provisioned cluster event notifications"

Amazon Redshift cluster event notification subscriptions

Amazon Redshift uses the Amazon Simple Notification Service (Amazon SNS) to communicate notifications of Amazon Redshift events. You enable notifications by creating an Amazon Redshift event subscription. You can be notified when an event occurs for a given cluster, snapshot, security group, or parameter group. The simplest way to create a subscription is with the Amazon SNS console. For information on creating an Amazon SNS topic and subscribing to it, see Getting started with Amazon SNS.

In the Amazon Redshift subscription, you specify a set of filters for Amazon Redshift events and an Amazon SNS topic. Whenever an event occurs that matches the filter criteria, Amazon Redshift publishes a notification message to the Amazon SNS topic.

Amazon SNS then transmits the message to any Amazon SNS consumers that have an Amazon SNS subscription to the topic. The messages sent to the Amazon SNS consumers can be in any form supported by Amazon SNS for an AWS Region, such as an email, a text message, or a call to an HTTP endpoint. For example, all Regions support email notifications, but SMS notifications can only be created in the US East (N. Virginia) Region.

Note

Currently, you can only create an event subscription to an Amazon SNS standard topic (not to an Amazon SNS FIFO topic). For more information, see Amazon SNS event sources in the *Amazon Simple Notification Service Developer Guide*.

When you create an event notification subscription, you specify one or more event filters. Amazon Redshift sends notifications through the subscription any time an event occurs that matches all of the filter criteria. The filter criteria include source type (such as cluster or snapshot), source ID (such as the name of a cluster or snapshot), event category (such as Monitoring or Security), and event severity (such as INFO or ERROR).

If you create event notification subscriptions using the CLI or API, you must create an Amazon Simple Notification Service topic and subscribe to that topic with the Amazon SNS console or Amazon SNS API. You will also need to retain the Amazon Resource Name (ARN) of the topic because it is used when submitting CLI commands or API actions.

You can easily turn off notification without deleting a subscription by setting the **Enabled** radio button to No in the AWS Management Console or by setting the `Enabled` parameter to `false` using the Amazon Redshift CLI or API.

An Amazon Redshift event subscription can specify these event criteria:

- Source type, the values are cluster, snapshot, parameter-groups, and security-groups.
- Source ID of a resource, such as `my-cluster-1` or `my-snapshot-20130823`. The ID must be for a resource in the same AWS Region as the event subscription.
- Event category, the values are Configuration, Management, Monitoring, Security, and Pending
- Event severity, the values are INFO or ERROR.

The event criteria can be specified independently, except that you must specify a source type before you can specify source IDs in the console. For example, you can specify an event category without having to specify a source type, source ID, or severity. While you can specify source IDs for resources that are not of the type specified in source type, no notifications will be sent for events from those resources. For example, if you specify a source type of cluster and the ID of a security group, none of the events raised by that security group would match the source type filter criteria, so no notifications would be sent for those events.

Amazon Redshift sends a notification for any event that matches all criteria specified in a subscription. Some examples of the sets of events returned:

- Subscription specifies a source type of cluster, a source ID of my-cluster-1, a category of Monitoring, and a severity of ERROR. The subscription will send notifications for only monitoring events with a severity of ERROR from my-cluster-1.
- Subscription specifies a source type of cluster, a category of Configuration, and a severity of INFO. The subscription will send notifications for configuration events with a severity of INFO from any Amazon Redshift cluster in the AWS account.
- Subscription specifies a category of Configuration, and a severity of INFO. The subscription will send notifications for configuration events with a severity of INFO from any Amazon Redshift resource in the AWS account.
- Subscription specifies a severity of ERROR. The subscription will send notifications for all events with a severity of ERROR from any Amazon Redshift resource in the AWS account.

If you delete or rename an object whose name is referenced as a source ID in an existing subscription, the subscription will remain active, but will have no events to forward from that object. If you later create a new object with the same name as is referenced in the subscription source ID, the subscription will start sending notifications for events from the new object.

Amazon Redshift publishes event notifications to an Amazon SNS topic, which is identified by its Amazon Resource Name (ARN). When you create an event subscription using the Amazon Redshift console, you can either specify an existing Amazon SNS topic, or request that the console create the topic when it creates the subscription.

All Amazon Redshift event notifications sent to the Amazon SNS topic are in turn transmitted to all Amazon SNS consumers that are subscribed to that topic. Use the Amazon SNS console to make changes to the Amazon SNS topic, such as adding or removing consumer subscriptions to the topic.

The following sections list all categories and events that you can be notified of. It also provides information about subscribing to and working with Amazon Redshift event subscriptions.

Creating an event notification subscription

You can create an Amazon Simple Notification Service (Amazon SNS) event notification subscription to send notifications when an event occurs for a given Amazon Redshift cluster, snapshot, security group, or parameter group. These notifications are sent to an SNS topic, which in turn transmits messages to any SNS consumers subscribed to the topic.

The SNS messages to the consumers can be in any notification form supported by Amazon SNS for an AWS Region, such as an email, a text message, or a call to an HTTP endpoint. For example, all regions support email notifications, but SMS notifications can only be created in the US East (N. Virginia) Region. For more information, see Amazon Redshift provisioned cluster event notifications.

To create an event subscription

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Events**.
3. Choose the **Event subscription** tab, then choose **Create event subscriptions**.
4. Enter the properties of your event subscription, such as name, source type, category, and severity. You can also enable Amazon SNS topics to get notified of events.
5. Choose **Create event subscriptions** to create your subscription.

Amazon Redshift provisioned cluster event notifications

This page shows the event IDs and categories for each Amazon Redshift source type.

Categories and events for the cluster source type

The following table shows the event category and a list of events when a cluster is the source type.

Amazon Redshift category	Event ID	Event severity	Description
Configuration	REDSHIFT-EVENT-1000	INFO	The parameter group [parameter group name] was updated at [time]. 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.
Configuration	REDSHIFT-EVENT-1001	INFO	Your Amazon Redshift cluster [cluster name] was modified to use parameter group [parameter group name] at [time].
Configuration	REDSHIFT-EVENT-1500	ERROR	The Amazon VPC [VPC name] does not exist. Your configuration changes for cluster [cluster name] were not applied. Please visit the AWS Management Console to correct the issue.
Configuration	REDSHIFT-EVENT-1501	ERROR	The customer subnets [subnet name] you specified for Amazon VPC [VPC name] do not exist or are invalid. Your configuration changes for cluster [cluster name] were not applied. Please visit the AWS Management Console to correct the issue.
Configuration	REDSHIFT-EVENT-1502	ERROR	Subnets in cluster subnet group [subnet group name] have no available IP addresses. Cluster [cluster name] could not be created.
Configuration	REDSHIFT-EVENT-1503	ERROR	The Amazon VPC [VPC name] has no internet gateway attached to it. Your configuration changes for cluster [cluster name] were not

Amazon Redshift category	Event ID	Event severity	Description
			applied. Please visit the AWS Management Console to correct the issue.
Configuration	REDSHIFT-EVENT-1504	ERROR	The HSM for cluster [cluster name] is unreachable.
Configuration	REDSHIFT-EVENT-1505	ERROR	The HSM for cluster [cluster name] cannot be registered. Try a different configuration.
Configuration	REDSHIFT-EVENT-1506	ERROR	Amazon Redshift exceeded your account's elastic network interface limit. Delete up to [maximum number of elastic network interfaces] elastic network interfaces or request a limit increase of the number of network interfaces per AWS Region with EC2.
Configuration	REDSHIFT-EVENT-1509	ERROR	The Amazon Redshift cluster [cluster name] can't be created because your account's VPC endpoint limit has been reached. Delete unused VPC endpoints or request an increase in the limit of VPC endpoints. For more information, see VPC endpoints in the *Amazon VPC User Guide*.
Configuration	REDSHIFT-EVENT-1510	ERROR	We have detected that the attempt to load sample data on your Amazon Redshift cluster [cluster name] didn't succeed. To load sample data, first configure your VPC to have access to Amazon S3 buckets, then create a new cluster and load sample data. For more information, see see Enabling enhanced VPC routing in the *Amazon Redshift Management Guide*.

Amazon Redshift category	Event ID	Event severity	Description
Configuration	REDSHIFT-EVENT-1511	ERROR	Amazon Redshift cluster [cluster name] cannot be created because you exceeded your account's limit of Elastic IP addresses. Delete unused Elastic IP addresses or request a limit increase with Amazon EC2.
Management	REDSHIFT-EVENT-2000	INFO	Your Amazon Redshift cluster: [cluster name] has been created and is ready for use.
Management	REDSHIFT-EVENT-2001	INFO	Your Amazon Redshift cluster [cluster name] was deleted at [time]. A final snapshot [was / was not] saved.
Management	REDSHIFT-EVENT-2002	INFO	VPC security groups for cluster [cluster name] updated at [time in UTC].
Management	REDSHIFT-EVENT-2003	INFO	Maintenance started on cluster [cluster name] at [time in UTC].
Management	REDSHIFT-EVENT-2004	INFO	Maintenance on cluster [cluster name] completed at [time in UTC].
Management	REDSHIFT-EVENT-2006	INFO	Cluster [cluster name] resize started at [time in UTC]. Cluster is in read-only mode.
Management	REDSHIFT-EVENT-2007	INFO	A resize request for cluster [cluster name] has been acknowledged.
Management	REDSHIFT-EVENT-2008	INFO	Your restore operation to create a new Amazon Redshift cluster [cluster name] snapshot [snapshot name] was started at [time]. To monitor restore progress, please visit the AWS Management Console.

Amazon Redshift category	Event ID	Event severity	Description
Management	REDSHIFT-EVENT-2013	INFO	Your Amazon Redshift cluster [cluster name] was renamed at [time].
Management	REDSHIFT-EVENT-2014	INFO	A table restore request for Amazon Redshift cluster [cluster name] has been received.
Management	REDSHIFT-EVENT-2015	INFO	Table restore was cancelled for Amazon Redshift cluster [cluster name] at [time].
Management	REDSHIFT-EVENT-2016	INFO	Replacement of your Amazon Redshift cluster [cluster name] was started at [time].
Management	REDSHIFT-EVENT-2017	INFO	Customer initiated maintenance started on your Amazon Redshift cluster [cluster name] at [time]. The cluster may not be available during maintenance.
Management	REDSHIFT-EVENT-2018	INFO	Customer initiated maintenance completed on your Amazon Redshift cluster [cluster name] at [time].
Management	REDSHIFT-EVENT-2019	ERROR	Customer initiated maintenance failed on your Amazon Redshift cluster [cluster name] at [time]. Returning the cluster back to its original state.
Management	REDSHIFT-EVENT-2020	INFO	Your Amazon Redshift cluster [cluster name]'s track has been modified from [from track] to [to track].

Amazon Redshift category	Event ID	Event severity	Description
Management	REDSHIFT-EVENT-2021	ERROR	The [operation] of Amazon Redshift cluster [cluster name] did not succeed while acquiring capacity from our capacity pool. We are working to acquire capacity but for now, we have cancelled your request. Delete this cluster and retry later.
Management	REDSHIFT-EVENT-2022	ERROR	The [operation] of Amazon Redshift cluster [cluster name] did not succeed while acquiring capacity from our capacity pool. We are working to acquire capacity but for now, we have cancelled your request. Capacity is available in [alternative Availability Zones]. Delete this cluster and retry in an alternative Availability Zone.
Management	REDSHIFT-EVENT-2023	ERROR	We have detected hardware failure on your single node Amazon Redshift cluster [cluster name], which may have resulted in failed queries or intermittent availability of the cluster. Replacing the cluster did not succeed while acquiring capacity from our capacity pool. You will need to restore a new cluster from a snapshot. Delete this cluster, select the latest available snapshot, and restore a new cluster from that snapshot. This will automatically provision you on healthy hardware.

Amazon Redshift category	Event ID	Event severity	Description
Management	REDSHIFT-EVENT-2024	ERROR	We have detected hardware failure on your single node Amazon Redshift cluster [cluster name], which may have resulted in failed queries or intermittent availability of the cluster. Replacing cluster did not succeed while acquiring capacity from our capacity pool. Capacity is available in Availability Zone: [alternative Availability Zones]. Delete this cluster, select the latest available snapshot, and restore a new cluster from that snapshot. This will automatically provision you on healthy hardware.
Management	REDSHIFT-EVENT-3011	INFO	Elastic resize for Amazon Redshift cluster '[cluster name]' started at [time]. We will hold the database connections during resize. Some queries and connections may be terminated or timed out during this operation.
Management	REDSHIFT-EVENT-3012	INFO	We have received an elastic resize request for the cluster '[cluster name]' started at [time]. We will provide an event notification when resize begins.
Pending	REDSHIFT-EVENT-2025	INFO	Your database for cluster [cluster name] will be updated between [start time] and [end time]. Your cluster will not be accessible. Plan accordingly.

Amazon Redshift category	Event ID	Event severity	Description
Pending	REDSHIFT-EVENT-2026	INFO	Your cluster [cluster name] will be updated between [start time] and [end time]. Your cluster will not be accessible. Plan according ly.
Monitoring	REDSHIFT-EVENT-2050	INFO	A hardware issue was detected on Amazon Redshift cluster [cluster name]. A replaceme nt request was initiated at [time].
Monitoring	REDSHIFT-EVENT-3000	INFO	Your Amazon Redshift cluster [cluster name] was rebooted at [time].
Monitoring	REDSHIFT-EVENT-3001	INFO	A node on your Amazon Redshift cluster: [cluster name] was automatically replaced at [time], and your cluster is operating normally.
Monitoring	REDSHIFT-EVENT-3002	INFO	The resize for your Amazon Redshift cluster [cluster name] is complete and your cluster is available for reads and writes. The resize was initiated at [time] and took [hours] hours to complete.
Monitoring	REDSHIFT-EVENT-3003	INFO	Amazon Redshift cluster [cluster name] was successfully created from snapshot [snapshot name] and is available for use.
Monitoring	REDSHIFT-EVENT-3007	INFO	Your Amazon Redshift snapshot [snapshot name] was copied successfully from [source AWS Region] to [destination AWS Region] at [time].
Monitoring	REDSHIFT-EVENT-3008	INFO	Table restore started for Amazon Redshift cluster [cluster name] at [time].

Amazon Redshift category	Event ID	Event severity	Description
Monitoring	REDSHIFT-EVENT-3009	INFO	Table restore completed successfully for Amazon Redshift cluster [cluster name] at [time].
Monitoring	REDSHIFT-EVENT-3010	ERROR	Table restore failed for Amazon Redshift cluster [cluster name] at [time].
Monitoring	REDSHIFT-EVENT-3013	ERROR	The requested elastic resize operation for Amazon Redshift cluster [cluster name] failed at [time] due to [reason].
Monitoring	REDSHIFT-EVENT-3014	INFO	Amazon Redshift rebooted cluster [cluster name] at [time].
Pending	REDSHIFT-EVENT-3015	INFO	We have scheduled a reboot for your cluster [cluster name] for maintenance between [start time in UTC] and [end time in UTC]. Your cluster won't be accessible during this time. We recommend that you plan for the outage to avoid any setbacks.
Monitoring	REDSHIFT-EVENT-3016	INFO	Cluster will be rebooted.
Monitoring	REDSHIFT-EVENT-3500	ERROR	The resize for your Amazon Redshift cluster [cluster name] failed. The resize will be automatically retried in a few minutes.
Monitoring	REDSHIFT-EVENT-3501	ERROR	Your restore operation to create Amazon Redshift cluster [cluster name] from snapshot [snapshot name] failed at [time]. Please retry your operation.
Monitoring	REDSHIFT-EVENT-3504	ERROR	The Amazon S3 bucket [bucket name] is not valid for logging for cluster [cluster name].

Amazon Redshift category	Event ID	Event severity	Description
Monitoring	REDSHIFT-EVENT-3505	ERROR	The Amazon S3 bucket [bucket name] does not have the correct IAM policies for cluster [cluster name].
Monitoring	REDSHIFT-EVENT-3506	ERROR	The Amazon S3 bucket [bucket name] does not exist. Logging cannot continue for cluster [cluster name].
Monitoring	REDSHIFT-EVENT-3507	ERROR	The Amazon Redshift cluster [cluster name] cannot be created using EIP [IP address]. This EIP is already in use.
Monitoring	REDSHIFT-EVENT-3508	ERROR	The Amazon Redshift cluster [cluster name] cannot be created using EIP [IP address]. The EIP cannot be found.
Monitoring	REDSHIFT-EVENT-3509	ERROR	Cross-region snapshot copy is not enabled for cluster [cluster name].
Monitoring	REDSHIFT-EVENT-3510	ERROR	Table restore failed to start for Amazon Redshift cluster [cluster name] at [time]. Reason: [reason].
Monitoring	REDSHIFT-EVENT-3511	ERROR	Table restore failed for Amazon Redshift cluster [cluster name] at [time].
Monitoring	REDSHIFT-EVENT-3512	ERROR	Amazon Redshift cluster [cluster name] has failed due to a hardware issue. The cluster is being automatically restored from the latest snapshot [snapshot name] created at [time].

Amazon Redshift category	Event ID	Event severity	Description
Monitoring	REDSHIFT-EVENT-3513	ERROR	Amazon Redshift cluster [cluster name] has failed due to a hardware issue. The cluster is being automatically restored from the latest snapshot [snapshot name] created at [time]. Any database changes made after this time will need to be resubmitted.
Monitoring	REDSHIFT-EVENT-3514	ERROR	Amazon Redshift cluster [cluster name] has failed due to a hardware issue. The cluster is being placed in hardware failure status. Please delete the cluster and restore from the latest snapshot [snapshot name] created at [time].
Monitoring	REDSHIFT-EVENT-3515	ERROR	Amazon Redshift cluster [cluster name] has failed due to a hardware issue. The cluster is being placed in hardware failure status. Please delete the cluster and restore from the latest snapshot [snapshot name] created at [time]. Any database changes made after this time will need to be resubmitted.
Monitoring	REDSHIFT-EVENT-3516	ERROR	Amazon Redshift cluster [cluster name] has failed due to a hardware issue and there are no backups for the cluster. The cluster is being placed in hardware failure status and can be deleted.
Monitoring	REDSHIFT-EVENT-3519	INFO	Cluster [cluster name] began restart at [time].
Monitoring	REDSHIFT-EVENT-3520	INFO	Cluster [cluster name] completed restart at [time].

Amazon Redshift category	Event ID	Event severity	Description
Monitoring	REDSHIFT-EVENT-3521	INFO	We detected a connectivity issue on the cluster '[cluster name]'. An automated diagnostics check has been initiated at [time].
Monitoring	REDSHIFT-EVENT-3522	INFO	Recovery action on '[cluster name]' cluster failed at [time]. The Amazon Redshift team is working on a solution.
Monitoring	REDSHIFT-EVENT-3533	ERROR	Cluster resize on '[cluster name]' was cancelled at [time]. The operation was cancelled because [reason]. [action needed].
Monitoring	REDSHIFT-EVENT-3534	INFO	The elastic resize for Amazon Redshift cluster '[cluster name]' completed at [time]. The cluster is now available for read and write operations while we transfer data. Some queries may take longer to finish until data transfer is complete.
Monitoring	REDSHIFT-EVENT-3537	INFO	Cluster '[cluster name]' data transfer completed at [time in UTC].
Monitoring	REDSHIFT-EVENT-3600	INFO	The requested resize operation for Amazon Redshift cluster '[cluster name]' was cancelled in the past. Rollback was completed at [time].
Pending	REDSHIFT-EVENT-3601	INFO	A node on your cluster [cluster name] will be replaced between [start time] and [end time]. You can't defer this maintenance. Plan accordingly.

Amazon Redshift category	Event ID	Event severity	Description
Pending	REDSHIFT-EVENT-3602	INFO	A node on your cluster [cluster name] is scheduled to be replaced between [start time] and [end time]. Your cluster will not be accessible. Plan accordingly.
Management	REDSHIFT-EVENT-3603	INFO	The restore operation to create cluster [cluster name] from snapshot [snapshot name] failed due to an internal error. The cluster is being placed in incompatible restore status and can be deleted. Try to restore the snapshot into a cluster with a different configuration.
Management	REDSHIFT-EVENT-3614	INFO	The scheduled action [scheduled action name] was created at [time in UTC]. The first invocation is scheduled at [time in UTC].
Management	REDSHIFT-EVENT-3615	INFO	The scheduled action [scheduled action name] is scheduled at [time in UTC].
Monitoring	REDSHIFT-EVENT-3616	INFO	The scheduled action [scheduled action name] at [time in UTC] finished with 'SUCCEEDED' status.
Monitoring	REDSHIFT-EVENT-3617	ERROR	The scheduled action [scheduled action name] was skipped at [time in UTC] due to delay.
Monitoring	REDSHIFT-EVENT-3618	INFO	The cluster [cluster name] pause operation started at [UTC time]. Pause Started
Monitoring	REDSHIFT-EVENT-3619	INFO	Amazon Redshift cluster [cluster name] was successfully paused at [UTC time].

Amazon Redshift category	Event ID	Event severity	Description
Management	REDSHIFT-EVENT-3626	INFO	The scheduled action [scheduled action name] was modified at [time in UTC]. The first invocation is scheduled at [time in UTC].
Management	REDSHIFT-EVENT-3627	INFO	The scheduled action [scheduled action name] was deleted at [time in UTC].
Monitoring	REDSHIFT-EVENT-3628	ERROR	The scheduled action [scheduled action name] at [time in UTC] finished with 'FAILED' status.
Monitoring	REDSHIFT-EVENT-3629	INFO	An internal failover request for cluster [cluster name] has been initiated.
Monitoring	REDSHIFT-EVENT-3630	INFO	Amazon Redshift successfully relocated your Amazon Redshift cluster [cluster name] from [availability-zone] to [availability-zone] for recovery.
Management	REDSHIFT-EVENT-3631	INFO	Amazon Redshift [cluster name] received your relocation request. When Availability Zone relocation completes, Amazon Redshift sends an event notification.
Monitoring	REDSHIFT-EVENT-3632	INFO	Amazon Redshift cluster [cluster name] was successfully relocated from [availability-zone] to [availability-zone]. You can use the cluster now.
Monitoring	REDSHIFT-EVENT-3658	ERROR	EC2-Classic to EC2-VPC migration failed for Redshift cluster [cluster id].
Monitoring	REDSHIFT-EVENT-3659	INFO	EC2-Classic to EC2-VPC migration succeeded for Redshift cluster [cluster id].

Amazon Redshift category	Event ID	Event severity	Description
Monitoring	REDSHIFT-EVENT-3660	INFO	The cluster is being placed in hardware failure status. Please delete the EC2-Classic cluster and restore to a EC2-VPC cluster from the latest snapshot [snapshot name] created at [time in UTC].
Management	REDSHIFT-EVENT-3666	INFO	Amazon Redshift Multi-AZ cluster [cluster name] has detected a failure at [time in UTC] and triggered an automatic recovery.
Management	REDSHIFT-EVENT-3667	INFO	Amazon Redshift Multi-AZ cluster [cluster name] successfully recovered at [time in UTC] and is available for use in [first availability zone]. Secondary compute in another AZ will be available shortly.
Monitoring	REDSHIFT-EVENT-3668	ERROR	Amazon Redshift Multi-AZ cluster [cluster name] has failed to recover at [time in UTC].
Management	REDSHIFT-EVENT-3669	INFO	Amazon Redshift Multi-AZ cluster [cluster name] successfully recovered at [time in UTC] and is available for use with compute resources from both [first availability zone] and [second availability-zone].
Management	REDSHIFT-EVENT-3670	INFO	Maintenance on Amazon Redshift cluster [cluster name] completed at [time in UTC] and is available for use with compute resources in [first availability zone]. Secondary compute in another AZ will be available shortly.

Amazon Redshift category	Event ID	Event severity	Description
Management	REDSHIFT-EVENT-3671	INFO	Resize on Amazon Redshift cluster [cluster name] completed at [time in UTC] and is available for use in [first availability zone]. Secondary compute in another AZ will be available shortly.
Management	REDSHIFT-EVENT-3672	INFO	Amazon Redshift Multi-AZ cluster [cluster name] has detected a failure in [second availability-zone] at [time in UTC] and triggered an automatic recovery.
Management	REDSHIFT-EVENT-3673	INFO	The operation to enable Multi-AZ for Amazon Redshift cluster [cluster name] has started at [time in UTC].
Management	REDSHIFT-EVENT-3674	INFO	The operation to enable Multi-AZ for Amazon Redshift cluster [cluster name] has successfully completed at [time in UTC].
Monitoring	REDSHIFT-EVENT-3675	ERROR	The operation to enable Multi-AZ for Amazon Redshift cluster [cluster name] has failed at [time in UTC].
Management	REDSHIFT-EVENT-3676	INFO	The operation to disable Multi-AZ for your Amazon Redshift Multi-AZ cluster [cluster name] has started at [time in UTC].
Management	REDSHIFT-EVENT-3677	INFO	The operation to disable Multi-AZ for your Amazon Redshift cluster [cluster name] has successfully completed at [time in UTC].
Monitoring	REDSHIFT-EVENT-3678	ERROR	The operation to disable Multi-AZ for your Amazon Redshift cluster [cluster name] has failed at [time in UTC].

Amazon Redshift category	Event ID	Event severity	Description
Configuration	REDSHIFT-EVENT-3679	INFO	The port of Amazon Redshift cluster [cluster name] modified successfully.
Configuration	REDSHIFT-EVENT-3680	ERROR	Amazon Redshift couldn't create cluster [cluster name] because the Service Linked Role (SLR) necessary for this operation is inaccessible. Try creating it again from the Amazon Redshift console. Amazon Redshift will create the SLR automatically.
Monitoring	REDSHIFT-EVENT-3684	ERROR	Your Amazon S3 bucket [bucket name] has been encrypted with an unknown or inaccessible AWS KMS key. Modify the encryption of your Amazon S3 bucket.
Management	REDSHIFT-EVENT-3685	ERROR	The restore operation on the cluster [cluster name] failed because it doesn't have enough disk space available. The operation is being rolled back. Try restoring to a cluster with a different configuration.
Management	REDSHIFT-EVENT-3686	ERROR	The resize operation on the cluster [cluster name] failed because it doesn't have enough disk space available. The operation is being rolled back. Try resizing to a cluster with a different configuration.
Management	REDSHIFT-EVENT-3687	INFO	The requested resize operation for Amazon Redshift cluster [cluster name] completed at [time in UTC].
Security	REDSHIFT-EVENT-3688	ERROR	The encryption key rotation for your Amazon Redshift cluster [cluster name] was not able to complete.

Amazon Redshift category	Event ID	Event severity	Description
DataSharing	REDSHIFT-EVENT-3689	INFO	The operation to register your Amazon Redshift namespace <cluster name> to Glue Data Catalog account <account id> was started at <time in UTC>.
DataSharing	REDSHIFT-EVENT-3690	INFO	The operation to register your Amazon Redshift namespace <cluster name> to Glue Data Catalog account <account id> has successfully completed at <time in UTC>.
DataSharing	REDSHIFT-EVENT-3691	INFO	The operation to register your Amazon Redshift namespace <cluster name> to Glue Data Catalog account <account id> has failed at <time in UTC>.
DataSharing	REDSHIFT-EVENT-3692	INFO	The operation to deregister your Amazon Redshift namespace <cluster name> from Glue Data Catalog account <account id> was started at <time in UTC>.
DataSharing	REDSHIFT-EVENT-3693	INFO	The operation to deregister your Amazon Redshift namespace <cluster name> from Glue Data Catalog account <account id> has successfully completed at <time in UTC>.
DataSharing	REDSHIFT-EVENT-3694	INFO	The operation to deregister your Amazon Redshift namespace <cluster name> from Glue Data Catalog account <account id> has failed at <time in UTC>.
Security	REDSHIFT-EVENT-4000	INFO	Your admin credentials for your Amazon Redshift cluster: [cluster name] were updated at [time].

Amazon Redshift category	Event ID	Event severity	Description
Security	REDSHIFT-EVENT-4001	INFO	The security group [security group name] was modified at [time]. The changes will take place for all associated clusters automatic ally.
Security	REDSHIFT-EVENT-4005	ERROR	Cluster [cluster name] has invalid AWS KMS key permissions. Your cluster will enter an "INACCESSIBLE_KMS_KEY" state and you will no longer be able to access it.
Security	REDSHIFT-EVENT-4006	INFO	Cluster [cluster name] has now been restored to its previous state because AWS KMS key permissions have been restored.
Security	REDSHIFT-EVENT-4500	ERROR	One or more of the security groups you provided for the Amazon Redshift cluster [cluster name] could not be found. If there were any valid security groups, they were applied to the cluster. Any invalid security groups were not applied. If you were modifying an existing cluster and no security groups were valid, the cluster kept its original security groups. If you were creating a new cluster and no security groups were valid, the cluster creation failed. In either case, please retry the request with valid security groups. For more information on managing security groups in Amazon Redshift, go to VPC security groups in the Amazon Redshift Management Guide.

Amazon Redshift category	Event ID	Event severity	Description
Security	REDSHIFT-EVENT-4501	ERROR	The security group [security group name] specified in Cluster Security Group [cluster security group name] could not be found. The authorization cannot be completed.
Security	REDSHIFT-EVENT-4502	ERROR	The admin credentials for Amazon Redshift cluster [cluster name] failed to update at [time] due to concurrent activity. Allow the current workload to complete or reduce the active workload and then retry the operation.
Security	REDSHIFT-EVENT-4503	ERROR	Amazon Redshift can't access the secret for your cluster [cluster name].
Security	REDSHIFT-EVENT-4504	ERROR	Amazon Redshift can't access the KMS key [KMS key] that was used to encrypt the admin credentials secret for your cluster [cluster name].
Security	REDSHIFT-EVENT-4505	ERROR	Amazon Redshift can't rotate the secret for your cluster [cluster name] because there's an ongoing operation on the cluster.
Security	REDSHIFT-EVENT-4506	ERROR	Your Amazon Redshift cluster [cluster name] is paused. Amazon Redshift can't rotate the secrets of paused clusters.

Categories and events for the parameter group source type

The following table shows the event category and a list of events when a parameter group is the source type.

Amazon Redshift category	Event ID	Event severity	Description
Configuration	REDSHIFT-EVENT-1002	INFO	The parameter [parameter name] was updated from [value] to [value] at [time].
Configuration	REDSHIFT-EVENT-1003	INFO	Cluster parameter group [group name] was created.
Configuration	REDSHIFT-EVENT-1004	INFO	Cluster parameter group [group name] was deleted.
Configuration	REDSHIFT-EVENT-1005	INFO	Cluster parameter group [name] was updated at [time]. 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.

Categories and events for the security group source type

The following tables shows the event category and a list of events when a security group is the source type.

Amazon Redshift category	Event ID	Event severity	Description
Security	REDSHIFT-EVENT-4002	INFO	Cluster security group [group name] was created.
Security	REDSHIFT-EVENT-4003	INFO	Cluster security group [group name] was deleted.

Amazon Redshift category	Event ID	Event severity	Description
Security	REDSHIFT-EVENT-4004	INFO	Cluster security group [group name] was changed at [time]. Changes will be automatically applied to all associated clusters.

Categories and events for the snapshot source type

The following tables shows the event category and a list of events when a snapshot is the source type.

Amazon Redshift category	Event ID	Event severity	Description
Management	REDSHIFT-EVENT-2009	INFO	A user snapshot [snapshot name] for Amazon Redshift Cluster [cluster name] started at [time]. To monitor snapshot progress, please visit the AWS Management Console.
Management	REDSHIFT-EVENT-2010	INFO	The user snapshot [snapshot name] for your Amazon Redshift cluster [cluster name] was cancelled at [time].
Management	REDSHIFT-EVENT-2011	INFO	The user snapshot [snapshot name] for Amazon Redshift cluster [cluster name] was deleted at [time].
Management	REDSHIFT-EVENT-2012	INFO	The final snapshot [snapshot name] for Amazon Redshift cluster [cluster name] was started at [time].

Amazon Redshift category	Event ID	Event severity	Description
Monitoring	REDSHIFT-EVENT-3004	INFO	The user snapshot [snapshot name] for your Amazon Redshift cluster [cluster name] completed successfully at [time].
Monitoring	REDSHIFT-EVENT-3005	INFO	The final snapshot [name] for Amazon Redshift cluster [name] completed successfully at [time].
Monitoring	REDSHIFT-EVENT-3006	INFO	The final snapshot [snapshot name] for Amazon Redshift cluster [cluster name] was cancelled at [time].
Monitoring	REDSHIFT-EVENT-3502	ERROR	The final snapshot [snapshot name] for Amazon Redshift cluster [cluster name] failed at [time]. The team is investigating the issue. Please visit the AWS Management Console to retry the operation.
Monitoring	REDSHIFT-EVENT-3503	ERROR	The user snapshot [snapshot name] for your Amazon Redshift cluster [cluster name] failed at [time]. The team is investigating the issue. Please visit the AWS Management Console to retry the operation.

Amazon Redshift Serverless event notifications with Amazon EventBridge

Amazon Redshift Serverless uses Amazon EventBridge to manage event notifications to keep you up-to-date regarding changes in your data warehouse. Amazon EventBridge is a serverless event bus service that you can use to connect your applications with data from a variety of sources.

In this case, the event source is Amazon Redshift. Events, which are monitored changes in an environment, are sent to EventBridge from your Amazon Redshift data warehouse automatically. Events are delivered in near-real time.

Capabilities of EventBridge include providing an environment for you to write event rules, which can specify actions to take for specific events. You can also set up targets, which are resources that EventBridge can send an event to. A target can include an API destination, an Amazon CloudWatch log group, and others. For more information about rules, see Amazon EventBridge rules. For more information about targets, see Amazon EventBridge targets.

Events can be classified into severities and categories. The following filters are available:

- *Resource filtering* – Receive messages based on the resource the events are associated with. Resources include a workgroup, a snapshot, and so on.
- *Time window filtering* – Scope events in a specific time period.
- *Category filtering* – Receive event notifications for all events in specified categories.

The following table includes Amazon Redshift Serverless events, with additional metadata:

Amazon Redshift Category	External Event ID	Event Severity	Message Description
RateChange	REDSHIFT-SERVERLESS-EVENT-1001	INFO	Workgroup base RPU change completed successfully at <time in UTC>.
RateChange	REDSHIFT-SERVERLESS-EVENT-1002	ERROR	Workgroup base RPU change failed to complete at <time in UTC>.
Monitoring	REDSHIFT-SERVERLESS-EVENT-1003	INFO	The software was updated on your Amazon Redshift Data Warehouse

Amazon Redshift Category	External Event ID	Event Severity	Message Description
			<endpoint name> at <time in UTC>.
Configuration	REDSHIFT-SERVERLESS-EVENT-1011	ERROR	Amazon Redshift Serverless couldn't create workgroup [workgroup name] because the Service Linked Role (SLR) necessary for this operation is inaccessible. Try creating it again on the Amazon Redshift console. Amazon Redshift will create the SLR automatically.
Monitoring	REDSHIFT-SERVERLESS-EVENT-1029	ERROR	Workgroup base RPU change failed to complete at [time in UTC] because it doesn't have enough disk space available. Try again with a different configuration.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Monitoring	REDSHIFT-SERVERLESS-EVENT-1500	ERROR	Workgroup <workgroup name> cannot be created or updated because you exceeded your account's limit of Elastic IP addresses . Delete unused Elastic IP addresses or request a limit increase with Amazon EC2.
Monitoring	REDSHIFT-SERVERLESS-EVENT-1501	ERROR	Subnet <subnet id> has no available IP addresses. This will prevent the following query types from running successfu lly on workgroup <workgroup name>: EMR, federated queries, COPY/UNLO AD from Amazon EC2. To correct the issue, free up IPs in your subnet by deleting ENIs.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Monitoring	REDSHIFT-SERVERLESS-EVENT-1502	ERROR	Subnet <subnet id> has no available IP addresses. This will prevent the Amazon EMR, Redshift federated queries, Redshift COPY/UNLOAD, Redshift ML query types from running successfully in workgroup <workgroup name>. To correct the issue, free up IPs in your subnet by deleting unused elastic network interfaces (ENIs).
Management	REDSHIFT-SERVERLESS-EVENT-1008	INFO	Your Amazon Redshift workgroup <workgroup name> has been created and is ready for use.
Management	REDSHIFT-SERVERLESS-EVENT-1009	INFO	Your Amazon Redshift workgroup <workgroup name> was deleted at <time in UTC>.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Monitoring	REDSHIFT-SERVERLESS-EVENT-1000	INFO	Snapshot <snapshot name> completed successfully at <time in UTC>.
Management	REDSHIFT-SERVERLESS-EVENT-1004	INFO	Restore from snapshot on namespace <namespace name> completed successfully at <time in UTC>.
Management	REDSHIFT-SERVERLESS-EVENT-1005	ERROR	Restore from snapshot on namespace <namespace name> failed at <time in UTC>.
Management	REDSHIFT-SERVERLESS-EVENT-1006	INFO	Restore from recovery point on namespace <namespace name> completed successfully at <time in UTC>.
Management	REDSHIFT-SERVERLESS-EVENT-1007	INFO	Restore from recovery point on namespace <namespace name> failed at <time in UTC>.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Security	REDSHIFT-SERVERLESS-EVENT-1012	ERROR	Amazon Redshift can't access the secret for your namespace <namespace name>.
Security	REDSHIFT-SERVERLESS-EVENT-1013	ERROR	Amazon Redshift can't access the KMS key that was used to encrypt the admin credentials secret for your namespace <namespace name>.
Security	REDSHIFT-SERVERLESS-EVENT-1014	ERROR	Amazon Redshift can't rotate the secret for your namespace <namespace name> because there's an ongoing operation on the workgroup.
Security	REDSHIFT-SERVERLESS-EVENT-1015	ERROR	Your namespace <namespace name> doesn't have a workgroup attached to it. Amazon Redshift can only rotate secrets for namespaces with workgroups attached to them.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Security	REDSHIFT-SERVERLESS-EVENT-1016	INFO	Admin credentials updated for your namespace <namespace name> at <time in UTC>.
Security	REDSHIFT-SERVERLESS-EVENT-1030	INFO	The operation to register your Amazon Redshift namespace <namespace name> to Glue Data Catalog account <account id> was started at <time in UTC>.
Security	REDSHIFT-SERVERLESS-EVENT-1031	INFO	The operation to register your Amazon Redshift namespace <namespace name> to Glue Data Catalog account <account id> has successfully completed at <time in UTC<.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Security	REDSHIFT-SERVERLESS-EVENT-1032	INFO	The operation to register your Amazon Redshift namespace <namespace name> to Glue Data Catalog account <account id> has failed at <time in UTC>..
Security	REDSHIFT-SERVERLESS-EVENT-1033	INFO	The operation to deregister your Amazon Redshift namespace <namespace name> from Glue Data Catalog account <account id> was started at <time in UTC>.
Security	REDSHIFT-SERVERLESS-EVENT-1034	INFO	The operation to deregister your Amazon Redshift namespace <namespace name> from Glue Data Catalog account <account id> has successfully completed at <time in UTC>.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Security	REDSHIFT-SERVERLESS-EVENT-1035	INFO	The operation to deregister your Amazon Redshift namespace <namespace name> from Glue Data Catalog account <account id> has failed at <time in UTC>.
Management	REDSHIFT-SERVERLESS-EVENT-1036	ERROR	The customer-initiated track update failed on your Redshift Serverless workgroup <workgroup name>. Returning the workgroup back to its original track.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Management	REDSHIFT-SERVERLESS-EVENT-1037	ERROR	The track update failed on your Redshift Serverless workgroup <workgroup name>. The track update failed for your Amazon Redshift Serverless workgroup because the workgroup was busy at the time. As a result, the workgroup has been returned to its original track. To retry the track update, please wait for a time of lower activity on the workgroup, and then attempt to update the track again.
Management	REDSHIFT-SERVERLESS-EVENT-1038	INFO	Your Amazon Redshift workgroup <workgroup name> track has been modified. The track change completed.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Namespace	REDSHIFT-SERVERLESS-EVENT-1039	ERROR	Namespace <namespace name> has invalid AWS KMS key permissions. Your namespace will enter an "INACCESSIBLE_KMS_KEY" state and you will no longer be able to access it.
Namespace	REDSHIFT-SERVERLESS-EVENT-1040	INFO	Namespace <namespace name> has now been restored to its previous state because AWS KMS key permissions have been restored.
Namespace	REDSHIFT-SERVERLESS-EVENT-1045	INFO	Datashare <datashare name> on your Redshift Serverless namespace <namespace name> was dropped due to serverless restore.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Namespace	REDSHIFT-SERVERLESS-EVENT-1046	INFO	Consumer <consumer principle id> for datashare <datashare name> on your Redshift Serverless namespace <namespace name> was revoked due to serverless restore.
Namespace	REDSHIFT-SERVERLESS-EVENT-1047	INFO	Public accessibility changed for datashare <datashare name> on your Redshift Serverless namespace <namespace name> due to serverless restore.
Security	REDSHIFT-SERVERLESS-EVENT-1048	ERROR	Your request to opt-out of the Redshift-Secrets Manager integration for namespace <namespace name> failed because of an internal issue. Retry the opt-out operation manually to resolve this issue.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Security	REDSHIFT-SERVERLESS-EVENT-1049	ERROR	Your request to opt-in to the Redshift-Secrets Manager integrati on for namespace <namespace name> could not complete successfully due to an internal issue. Rotate the secret associated with your Redshift Serverless namespace manually to resolve this issue.
Security	REDSHIFT-SERVERLESS-EVENT-1050	ERROR	Your request to reset the admin credentia ls on namespace <namespace name> failed because of an internal issue. Retry the operation to reset the admin user credentials.

Zero-ETL integration event notifications with Amazon EventBridge

Zero-ETL integration uses Amazon EventBridge to manage event notifications to keep you up-to-date regarding changes in your integrations. Amazon EventBridge is a serverless event bus service that you can use to connect your applications with data from a variety of sources. In this case, the event source is Amazon Redshift. Events, which are monitored changes in an environment, are sent

to EventBridge from your Amazon Redshift data warehouse automatically. Events are delivered in near real time.

EventBridge provides an environment for you to write event rules, which can specify actions to take for specific events. You can also set up targets, which are resources that EventBridge can send an event to. A target can include an API destination, an Amazon CloudWatch log group, and others. For more information about rules, see Amazon EventBridge rules. For more information about targets, see Amazon EventBridge targets.

Events can be classified into severities and categories. The following filters are available:

- *Resource filtering* – Receive messages based on the resource that the events are associated with. Resources include a workgroup or a snapshot.
- *Time window filtering* – Scope events in a specific time period.
- *Category filtering* – Receive event notifications for all events in specified categories.

The following table includes zero-ETL integration events, with additional metadata:

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Monitoring	REDSHIFT-INTEGRATION-EVENT-0000	INFO	Your zero-ETL Integration <integration name> has been created and became ACTIVE at <time in UTC>.
Monitoring	REDSHIFT-INTEGRATION-EVENT-0001	INFO	Your zero-ETL Integration <integration name> has been deleted at <time in UTC>.
Monitoring	REDSHIFT-INTEGRATION-EVENT-0002	INFO	Your zero-ETL Integration <integration name> has

Amazon Redshift Category	External Event ID	Event Severity	Message Description
			initiated deletion at <time in UTC>.
Monitoring	REDSHIFT-INTEGRATION-EVENT-0003	INFO	Your zero-ETL Integration <integration name> is synchronizing transactional data to the Amazon Redshift data warehouse.
Monitoring	REDSHIFT-INTEGRATION-EVENT-0004	WARNING	Amazon Redshift cannot replicate the table because the table is missing a primary key. Add primary key(s) to the source table and the table will resynchronize automatically.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Monitoring	REDSHIFT-INTEGRATION-EVENT-0005	WARNING	Amazon Redshift cannot replicate the table because one or more columns use an unsupported data type. Modify your integration to exclude this table in the filter, or drop column(s) from source table and run 'ALTER DATABASE <Redshift database name> INTEGRATION REFRESH TABLE <schema name>.<table name>' to synchronize this table.
Monitoring	REDSHIFT-INTEGRATION-EVENT-0006	ERROR	Unable to create integration. Delete and recreate the integration. If the error persists, contact AWS Support.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Monitoring	REDSHIFT-INTEGRATION-EVENT-0007	ERROR	Unable to load data due to an internal error. Delete and recreate the integration. If the error persists, contact AWS Support.
Monitoring	REDSHIFT-INTEGRATION-EVENT-0008	ERROR	Authorization failed because permissions have been revoked from the source DB cluster. Check the KMS key permission if you're using a customer-managed key (CMK) to encrypt the integration. Then, delete and recreate the integration.
Monitoring	REDSHIFT-INTEGRATION-EVENT-0009	ERROR	Unable to send data to Amazon Redshift because the number of tables and schemas exceed the Amazon Redshift limit. Delete and recreate the integration.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Monitoring	REDSHIFT-INTEGRATION-EVENT-0012	ERROR	A restore from recovery point was invoked on the destination serverless namespace. Delete and recreate the integration.
Monitoring	REDSHIFT-INTEGRATION-EVENT-0013	INFO	Your zero-ETL Integration <integration name> is now ACTIVE.
Monitoring	REDSHIFT-INTEGRATION-EVENT-0014	ERROR	Your integration <integration name> is in FAILED state because we are unable to modify it due to an internal error. Delete and recreate the integration. If the error persists, contact AWS Support.
Operation	REDSHIFT-INTEGRATION-EVENT-0016	INFO	Your zero-ETL integration <integration name> is processing a modification request.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Operation	REDSHIFT-INTEGRATION-EVENT-0017	INFO	Your modification to zero-ETL integration <integration name> has been applied.
Operation	REDSHIFT-INTEGRATION-EVENT-0018	WARNING	Target Amazon Redshift cluster is being paused. Wait for the cluster to be paused and then resume the cluster to continue streaming data.
Operation	REDSHIFT-INTEGRATION-EVENT-0019	WARNING	Target Amazon Redshift cluster is paused. The cluster must be resumed in order to continue streaming data.
Operation	REDSHIFT-INTEGRATION-EVENT-0020	WARNING	Target Amazon Redshift cluster is being resumed. Wait for the cluster to be active to continue streaming data.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Monitoring	REDSHIFT-INTEGRATION-EVENT-0025	ERROR	A serverless restore was invoked on the destination serverless namespace, and your integration is deactivated because it's not available in the restored snapshot. Restore a snapshot which is created when the integration is available, or delete the integration.
Configuration	REDSHIFT-INTEGRATION-EVENT-1000	ERROR	One or more parameters on the source Aurora DB cluster are misconfigured. Fix the parameter group and reboot the cluster to apply the changes, then recreate the integration.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Configuration	REDSHIFT-INTEGRATION-EVENT-1001	ERROR	Integration failed because the value of the enable_case_sensitive_identifier parameter is incorrect. Set the value to true for the source Aurora DB cluster, then delete and recreate the integration.
Configuration	REDSHIFT-INTEGRATION-EVENT-1002	ERROR	Integration failed because the value of the cdc_insert_enabled parameter is incorrect. Set the value to true for the source Aurora DB cluster, then delete and recreate the integration.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Configuration	REDSHIFT-INTEGRATION-EVENT-1003	ERROR	The binlog_format parameter in the source DB cluster parameter group must be set to ROW. Fix the parameter group and reboot the cluster to apply the change, then recreate the integration.
Configuration	REDSHIFT-INTEGRATION-EVENT-1004	ERROR	Unable to load data because the binlog_transaction_compression cluster parameter is enabled. Set the parameter value to OFF and reboot the writer instance to apply the change, then recreate the integration.

Amazon Redshift Category	External Event ID	Event Severity	Message Description
Configuration	REDSHIFT-INTEGRATION-EVENT-1005	ERROR	Unable to load data because the binlog_row_value_options cluster parameter is set to PARTIAL_JSON, which is not supported. Fix the parameter group and reboot the writer instance to apply the change, then recreate the integration.
Configuration	REDSHIFT-INTEGRATION-EVENT-1006	WARNING	Unable to parse integration filter. Fix the filter syntax.

Quotas and limits in Amazon Redshift

Amazon Redshift has quotas that limit the use of several resources in your AWS account per AWS Region. There is a default value for each quota and some quotas are adjustable.

Quotas for Amazon Redshift objects

Amazon Redshift has quotas that limit the use of several object types. There is a default value for each.

Quota name	AWS default value	Adjustable	Description
AWS accounts that you can authorize to restore a snapshot per snapshot	20	No	The maximum number of AWS accounts that you can authorize to restore a snapshot, per snapshot.
AWS accounts that you can authorize to restore a snapshot per AWS KMS key	100	No	The maximum number of AWS accounts that you can authorize to restore a snapshot, per KMS key. That is, if you have 10 snapshots that are encrypted with a single KMS key, then you can authorize 10 AWS accounts to restore each snapshot, or other combinations that add up to 100 accounts and do not exceed 20 accounts for each snapshot.
Cluster IAM	50[1]	No	The maximum number of IAM roles that you can associate with a cluster to authorize Amazon Redshift

Quota name	AWS default value	Adjustable	Description
roles for Amazon Redshift to access other AWS services			to access other AWS services for the user that owns the cluster and IAM roles. [1]The quota is 10 in the following AWS Regions: us-iso-east-1, us-iso-west-1, us-isob-east-1.
Concurrency level (query slots) for all user-defined manual WLM queues	50	No	The maximum query slots for all user-defined queues defined by manual workload management.
Concurrency scaling clusters	10	Yes	The maximum number of concurrency scaling clusters.
DC2 nodes in a cluster	128	Yes	The maximum number of DC2 nodes that you can allocate to a cluster. For more information about node limits for each node type, see Clusters and nodes in Amazon Redshift.
Event subscriptions	20	Yes	The maximum number of event subscriptions for this account in the current AWS Region.
Nodes	200	Yes	The maximum number of nodes across all database instances for this account in the current AWS Region.

Quota name	AWS default value	Adjustable	Description
Parameter groups	20	No	The maximum number of parameter groups for this account in the current AWS Region.
RG nodes in a cluster	128	Yes	The maximum number of RG nodes that you can allocate to a cluster. For more information about node limits for each node type, see Clusters and nodes in Amazon Redshift.
RA3 nodes in a cluster	128	Yes	The maximum number of RA3 nodes that you can allocate to a cluster. For more information about node limits for each node type, see Clusters and nodes in Amazon Redshift.
Redshift-managed VPC endpoints connected to a cluster	30	Yes	The maximum number of Redshift-managed VPC endpoints that you can connect to a cluster. For more information about Redshift-managed VPC endpoints, see Redshift-managed VPC endpoints.
Grantees to cluster accessed through a Redshift-managed VPC endpoint	10	Yes	The maximum number of grantees that a cluster owner can authorize to create a Redshift-managed VPC endpoint for a cluster. For more information about Redshift-managed VPC endpoints, see Redshift-managed VPC endpoints.

Quota name	AWS default value	Adjustable	Description
Redshift-managed VPC endpoints per authorization	10	Yes	The maximum number of Redshift-managed VPC endpoints that you can create per authorization. For more information about Redshift-managed VPC endpoints, see Redshift-managed VPC endpoints.
Reserved nodes	200	Yes	The maximum number of reserved nodes for this account in the current AWS Region.
Schemas in each database per cluster	9,900	No	The maximum number of schemas that you can create in each database, per cluster. However, `pg_temp_*` schemas do not count towards this quota.
Security groups	20	Yes	The maximum number of security groups for this account in the current AWS Region.
Single row size when loading by COPY	64	No	The maximum size (in MB) of a single row when loading by using the COPY command. This is the maximum size of a single unparsed record that the COPY command can scan; an individual SUPER object within the row is independently limited to 16 MB.
Snapshots	700	Yes	The maximum number of user snapshots for this account in the current AWS Region.
Subnet groups	50	Yes	The maximum number of subnet groups for this account in the current AWS Region.
Subnets in a subnet group	20	Yes	The maximum number of subnets for a subnet group.

Quota name	AWS default value	Adjustable	Description
Tables for `rg.large` cluster node type with a single-node cluster	9,900	No	The maximum number of tables for the `rg.large` cluster node type with a single-node cluster. This limit includes permanent tables, temporary tables, datashare tables, and materialized views. External tables are counted as temporary tables. Temporary tables include user-defined temporary tables and temporary tables created by Amazon Redshift during query processing or system maintenance. Views and system tables aren't included in this limit.
Tables for `rg.large` cluster node type with a multi-node cluster	20,000	No	The maximum number of tables for the `rg.large` node type with a multi-node cluster. This limit includes permanent tables, temporary tables, datashare tables, and materialized views. External tables are counted as temporary tables. Temporary tables include user-defined temporary tables and temporary tables created by Amazon Redshift during query processing or system maintenance. Views and system tables aren't included in this limit.
Tables for `rg.xlarge` cluster node type.	20,000	No	The maximum number of tables for the `rg.xlarge` cluster node type. This limit includes permanent tables, temporary tables, datashare tables, and materialized views. External tables are counted as temporary tables. Temporary tables include user-defined temporary tables and temporary tables created by Amazon Redshift during query processing or system maintenance. Views and system tables aren't included in this limit.

Quota name	AWS default value	Adjustable	Description
Tables for `rg.4xlarge` cluster node type	200,000	No	The maximum number of tables for the `rg.4xlarge` cluster node type. This limit includes permanent tables, temporary tables, datashare tables, and materialized views. External tables are counted as temporary tables. Temporary tables include user-defined temporary tables and temporary tables created by Amazon Redshift during query processing or system maintenance. Views and system tables aren't included in this limit.
Tables for `rg.12xlarge` cluster node type	200,000	No	The maximum number of tables for the `rg.12xlarge` cluster node type. This limit includes permanent tables, temporary tables, datashare tables, and materialized views. External tables are counted as temporary tables. Temporary tables include user-defined temporary tables and temporary tables created by Amazon Redshift during query processing or system maintenance. Views and system tables aren't included in this limit.
Tables for `ra3.large` cluster node type	9,900	No	The maximum number of tables for the `ra3.large` cluster node type. This limit includes permanent tables, temporary tables, datashare tables, and materialized views. External tables are counted as temporary tables. Temporary tables include user-defined temporary tables and temporary tables created by Amazon Redshift during query processing or system maintenance. Views and system tables aren't included in this limit.

Quota name	AWS default value	Adjustable	Description
Tables for `ra3.xlplus` cluster node type with a single-node cluster.	9,900	No	The maximum number of tables for the `ra3.xlplus` cluster node type with a single-node cluster. This limit includes permanent tables, temporary tables, datashare tables, and materialized views. External tables are counted as temporary tables. Temporary tables include user-defined temporary tables and temporary tables created by Amazon Redshift during query processing or system maintenance. Views and system tables aren't included in this limit.
Tables for `ra3.xlplus` cluster node type with a multiple-node cluster.	20,000	No	The maximum number of tables for the `ra3.xlplus` cluster node type with a multiple-node cluster. This limit includes permanent tables, temporary tables, datashare tables, and materialized views. External tables are counted as temporary tables. Temporary tables include user-defined temporary tables and temporary tables created by Amazon Redshift during query processing or system maintenance. Views and system tables aren't included in this limit.
Tables for `ra3.4xlarge` cluster node type	200,000	No	The maximum number of tables for the `ra3.4xlarge` cluster node type. This limit includes permanent tables, temporary tables, datashare tables, and materialized views. External tables are counted as temporary tables. Temporary tables include user-defined temporary tables and temporary tables created by Amazon Redshift during query processing or system maintenance. Views and system tables aren't included in this limit.

Quota name	AWS default value	Adjustable	Description
Tables for `ra3.16xlarge` cluster node type	200,000	No	The maximum number of tables for the `ra3.16xlarge` cluster node type. This limit includes permanent tables, temporary tables, datashare tables, and materialized views. External tables are counted as temporary tables. Temporary tables include user-defined temporary tables and temporary tables created by Amazon Redshift during query processing or system maintenance. Views and system tables aren't included in this limit.
Tables for `dc2.large` cluster node type	9,900	No	The maximum number of tables for the `dc2.large` cluster node type. This limit includes permanent tables, temporary tables, datashare tables, and materialized views. External tables are counted as temporary tables. Temporary tables include user-defined temporary tables and temporary tables created by Amazon Redshift during query processing or system maintenance. Views and system tables aren't included in this limit.
Tables for `dc2.8xlarge` cluster node type	200,000	No	The maximum number of tables for the `dc2.8xlarge` cluster node type. This limit includes permanent tables, temporary tables, datashare tables, and materialized views. External tables are counted as temporary tables. Temporary tables include user-defined temporary tables and temporary tables created by Amazon Redshift during query processing or system maintenance. Views and system tables aren't included in this limit.
Number of databases	60	No	The maximum allowed count of databases in an Amazon Redshift cluster. This excludes databases created from datashares.

Quota name	AWS default value	Adjustable	Description
Timeout for idle or inactive sessions	4 hours	No	This setting applies to the cluster. For information about setting the idle-session timeout value for a user, see ALTER USER in the *Amazon Redshift Database Developer Guide*. The user setting takes precedence over the cluster setting.
Timeout for idle transactions	6 hours	No	The maximum period of inactivity for an open transaction before Amazon Redshift ends the session associated with the transaction. This setting takes precedence over any user-defined idle timeout setting. It applies to the cluster.
Stored procedures in a database	10,000	No	The maximum number of stored procedures. See Limits and differences for stored procedure support for more limits.
Maximum number of connections for RG nodes	2,000	No	The maximum number of connections to an RG cluster. This is a hard limit that applies to all RG node types and can't be increased.
Maximum number of connections for RA3 nodes	2,000	No	The maximum number of connections to an RA3 cluster. This is a hard limit that applies to all RA3 node types and can't be increased.
Maximum number of connections for DC2 nodes	Varies	No	The maximum number of connections to a dc2.large cluster is 500. The maximum number of connections to a dc2.8xlarge cluster is 2000.

Quota name	AWS default value	Adjustable	Description
Number of Amazon Redshift roles in a cluster	1,000	Yes	The maximum number of Amazon Redshift roles that you can create per cluster. For more informati on about role-based access control (RBAC) roles see Role-based access control (RBAC) in the *Amazon Redshift Database Developer Guide*

Quotas for Amazon Redshift Serverless objects

Amazon Redshift has quotas that limit the use of several object types in your Amazon Redshift Serverless namespace. There is a default value for each.

Quota name	AWS default value	Adjustable	Description
Number of databases	100	No	The maximum allowed count of databases in an Amazon Redshift Serverless namespace. This excludes databases created from datashares.
Number of schemas	9,900	No	The maximum allowed count of schemas in an Amazon Redshift Serverless namespace.
Number of tables	200,000	No	The maximum allowed count of tables in an Amazon Redshift Serverless namespace.
Redshift Managed storage units for data warehouse s using 4 Redshift	up to 32 TB	No	The maximum RMS storage supported for a data warehouse with 4 RPUs. Within the 32 TB limit, Amazon Redshift uses a few hundred MBs of storage internally.

Quota name	AWS default value	Adjustable	Description
Processing Units (RPUs)			
Redshift Managed storage units for data warehouses using 8, 16, or 24 Redshift Processing Units (RPUs)	up to 128 TB	No	The maximum RMS storage supported for a data warehouse with 8, 16, or 24 RPUs. Within the 128 TB limit, Amazon Redshift uses a few hundred MBs of storage internally.
Timeout for idle or inactive sessions	1 hour	No	For information about setting the idle-session timeout value for a user, see ALTER USER in the *Amazon Redshift Database Developer Guide*. The user setting takes precedence.
Timeout for a running query	86,399 seconds (24 hours)	No	The maximum time for a running query before Amazon Redshift ends it.
Timeout for idle transactions	6 hours	No	The maximum period of inactivity for an open transaction before Amazon Redshift Serverless ends the session associated with the transaction. This setting takes precedence over any user-defined idle timeout setting.

Quota name	AWS default value	Adjustable	Description
Number of maximum connections	2000	No	The maximum number of connections allowed to connect to a workgroup.
Number of workgroups	25	Yes	The number of workgroups supported.
Number of namespaces	25	Yes	The number of namespaces supported.
Number of Amazon Redshift roles in a workgroup	1,000	Yes	The maximum number of Amazon Redshift roles that you can create per workgroup. For more information about role-based access control (RBAC) roles see Role-based access control (RBAC) in the *Amazon Redshift Database Developer Guide*
Base Redshift Processing Units (RPUs) per AWS account	The greater of either 3,200 RPUs or 1.5 times your maximum aggregate base RPUs from the previous six months.	Yes	The AWS account quota for Redshift Serverless base RPUs is the greater of either 3,200 RPUs or 1.5 times your maximum aggregate base RPUs from the previous six months. .

For more information about how Amazon Redshift Serverless billing is affected by timeout configuration, see Billing for Amazon Redshift Serverless.

Quotas for Amazon Redshift Data API

Amazon Redshift has quotas that limit the use of the Redshift Data API. There is a default value for each. For more information about Amazon Redshift Data API, see Using the Amazon Redshift Data API.

Quota name	AWS default value	Adjustable	Description
Transactions per second (TPS) for the `BatchExecuteStatement` API	20	No	The maximum number of operation requests you can make per second without being throttled.
Transactions per second (TPS) for the `CancelStatement` API	3	No	The maximum number of operation requests you can make per second without being throttled.
Transactions per second (TPS) for the `DescribeS`	100	No	The maximum number of operation requests you can make per second without being throttled.

Quota name	AWS default value	Adjustable	Description
tatement API			
Transactions per second (TPS) for the DescribeTable API	3	No	The maximum number of operation requests you can make per second without being throttled.
Transactions per second (TPS) for the ExecuteStatement API	30	No	The maximum number of operation requests you can make per second without being throttled.
Transactions per second (TPS) for the GetStatementResult API	20	No	The maximum number of operation requests you can make per second without being throttled.

Quota name	AWS default value	Adjustable	Description
Transactions per second (TPS) for the `ListDatabases` API	3	No	The maximum number of operation requests you can make per second without being throttled.
Transactions per second (TPS) for the `ListSchemas` API	3	No	The maximum number of operation requests you can make per second without being throttled.
Transactions per second (TPS) for the `ListStatements` API	3	No	The maximum number of operation requests you can make per second without being throttled.
Transactions per second (TPS) for the `ListTables` API	3	No	The maximum number of operation requests you can make per second without being throttled.

Quotas for query editor v2 objects

Amazon Redshift has quotas that limit the use of several object types in your Amazon Redshift query editor v2. There is a default value for each.

Quota name	AWS default value	Adjustable	Description
Connections	500	Yes	Maximum number of connections that you can create using the query editor v2 in this account in the current Region.
Active principals per account	50	Yes	Maximum number of simultaneous principals who can use query editor v2 in this account in the current Region.
Saved queries	2,500	Yes	Maximum number of saved queries that you can create using the query editor v2 in this account in the current Region.
Query versions	20	Yes	Maximum number of versions per query that you can create using the query editor v2 in this account in the current Region.
Saved charts	500	Yes	Maximum number of saved charts that you can create using the query editor v2 in this account in the current Region.
Data fetched size per query	100	No	Maximum size, in megabytes, of the data fetched per query by the query editor v2 in this account in the current Region.
Data fetched	5	No	Maximum size, in megabytes, of the data fetched per one query results page (100 rows) by the query editor v2 in this account and current Region.

Quota name	AWS default value	Adjustable	Description
size per 100 rows			
Maximum concurrent connections	3	No	Maximum database connections per user (includes isolated sessions). This value can be set from 1–10 by the query editor v2 administrator in **Account settings**. If you reach the limit set by your administrator, consider using shared sessions instead of isolated sessions when running your SQL. For more information about connections, see Opening query editor v2. For more information about setting the limit, see Account settings.

Quotas and limits for Amazon Redshift Spectrum objects

Amazon Redshift Spectrum has the following quotas and limits:

- The maximum number of databases per AWS account when using an AWS Glue Data Catalog. For this value, see AWS Glue service quotas in the *Amazon Web Services General Reference*.
- The maximum number of tables per database when using an AWS Glue Data Catalog. For this value, see AWS Glue service quotas in the *Amazon Web Services General Reference*.
- The maximum number of partitions per table when using an AWS Glue Data Catalog. For this value, see AWS Glue service quotas in the *Amazon Web Services General Reference*.
- The maximum number of partitions per AWS account when using an AWS Glue Data Catalog. For this value, see AWS Glue service quotas in the *Amazon Web Services General Reference*.
- The maximum number of columns for external tables when using an AWS Glue Data Catalog, 1,597 when pseudocolumns are enabled, and 1,600 when pseudocolumns aren't enabled.
- The maximum size of a string value in an ION or JSON file when using an AWS Glue Data Catalog is 16 KB. The string can be truncated if you reach this limit.
- You can add a maximum of 100 partitions using a single ALTER TABLE statement.
- All S3 data must be located in the same AWS Region as the Amazon Redshift cluster.

- Timestamps in ION and JSON must use ISO8601 format.
- External compression of ORC files is not supported.
- Text, OpenCSV, and Regex SERDEs do not support octal delimiters larger than '\177'.
- You must specify a predicate on the partition column to avoid reads from all partitions.

  For example, the following predicate filters on the column `ship_dtm`, but doesn't apply the filter to the partition column `ship_yyyymm`:

  ```
  WHERE ship_dtm > '2018-04-01'.
  ```

  To skip unneeded partitions you need to add a predicate `WHERE ship_yyyymm = '201804'`. This predicate limits read operations to the partition `\ship_yyyymm=201804\`.

These limits don't apply to an Apache Hive metastore.

Naming constraints

The following table describes naming constraints within Amazon Redshift.

Cluster identifier	• A cluster identifier must contain only lowercase characters. • It must contain 1–63 alphanumeric characters or hyphens. • Its first character must be a letter. • It cannot end with a hyphen or contain two consecutive hyphens. • It must be unique for all clusters within an AWS account.
Database name	• A database name must contain 1–64 alphanumeric characters. •

	It must contain only lowercase letters. • It cannot be a reserved word. For a list of reserved words, see Reserved words in the *Amazon Redshift Database Developer Guide*.
Endpoint name of a Redshift-managed VPC endpoint	• An endpoint name must contain 1–30 characters. • Valid characters are A-Z, a-z, 0-9, and hyphen(-). • The first character must be a letter. • The name can't contain two consecutive hyphens or end with a hyphen.
Admin user name	• An admin user name must contain only lowercase characters. • It must contain 1–128 alphanumeric characters. • Its first character must be a letter. • It cannot be a reserved word. For a list of reserved words, see Reserved words in the *Amazon Redshift Database Developer Guide*.

Admin password	• An admin password must contain 8–64 characters. • It must contain at least one uppercase letter. • It must contain at least one lowercase letter. • It must contain one number. • It can use any ASCII characters with ASCII codes 33–126, except ' (single quote), " (double quote), \, /, or @.
Parameter group name	• A parameter group name must contain 1–255 alphanumeric characters or hyphens. • It must contain only lowercase characters. • Its first character must be a letter. • It can't end with a hyphen or contain two consecutive hyphens.
Cluster security group name	• A cluster security group name must contain no more than 255 alphanumeric characters or hyphens. • It must contain only lowercase characters. • It must not be **`Default`**. • It must be unique for all security groups that are created by your AWS account.

Subnet group name	• A subnet group name must contain no more than 255 alphanumeric characters or hyphens. • It must contain only lowercase characters. • It must not be **`Default`**. • It must be unique for all subnet groups that are created by your AWS account.
Cluster snapshot identifier	• A cluster snapshot identifier must contain no more than 255 alphanumeric characters or hyphens. • It must contain only lowercase characters. • It must not be **`Default`**. • It must be unique for all snapshot identifiers that are created by your AWS account.

Tag resources in Amazon Redshift

In AWS, tags are user-defined labels that consist of key-value pairs. Amazon Redshift supports tagging to provide metadata about resources at a glance, and to categorize your billing reports based on cost allocation. To use tags for cost allocation, you must first activate those tags in the AWS Billing and Cost Management service. For more information about setting up and using tags for billing purposes, see Use cost allocation tags for custom billing reports and Setting up your monthly cost allocation report.

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.

Each resource has one *tag set*, which is a collection of one or more tags assigned to the resource. Each resource can have up to 50 tags per tag set. You can add tags when you create a resource and after a resource has been created. You can add tags to the following resource types in Amazon Redshift:

- CIDR/IP
- Cluster
- Cluster security group
- Cluster security group ingress rule
- Amazon EC2 security group
- Hardware security module (HSM) connection
- HSM client certificate
- Parameter group

- Snapshot
- Subnet group
- Integration (zero-ETL integration)

To use tagging from the Amazon Redshift console, your user can attach the AWS managed policy `AmazonRedshiftFullAccess`. For an example IAM policy with limited tagging permissions that you can attach to an Amazon Redshift console user, see Example 7: Allow a user to tag resources with the Amazon Redshift console. For more information about tagging, see What is AWS Resource Groups?.

Tagging requirements

Tags have the following requirements:

- Keys can't be prefixed with `aws:`.
- Keys must be unique per tag set.
- A key must be between 1 and 128 allowed characters.
- A value must be between 0 and 256 allowed characters.
- Values do not need to be unique per tag set.
- Allowed characters for keys and values are Unicode letters, digits, white space, and any of the following symbols: _ . : / = + - @.
- Keys and values are case sensitive.

Managing resource tags

The following procedure walks you through how to work with tags of your resources.

To manage tags on your Amazon Redshift resources

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.
2. On the navigation menu, choose **Configurations**, then choose **Manage tags**.
3. Enter your choices for the resources and choose which tags to add, modify, or delete. Then choose **Manage tags of the resources that you chose**.

Resources that you can tag include clusters, parameter groups, subnet groups, HSM client certificates, HSM connections, and snapshots.

4. On the **Manage tags** navigation page, choose **Review and apply tag changes**, then choose **Apply** to save your changes.

AWS Backup integration with Amazon Redshift

AWS Backup is a fully managed service that helps you centralize and automate data protection across AWS services, in the cloud, and on premises.

Using AWS Backup for Amazon Redshift, you can configure data protection policies and monitor activity for different Amazon Redshift resources in one place. You can also create and store snapshots on Amazon Redshift provisioned clusters and serverless namespaces. This lets you automate and consolidate backup tasks that you had to do separately before, without any manual processes.

Note

No-backup tables aren't supported for RG or RA3 provisioned clusters and Amazon Redshift Serverless workgroups. A table marked as no-backup in an RG or RA3 cluster or serverless workgroup is treated as a permanent table that will always be backed up while taking a snapshot, and always restored when restoring from a snapshot. To avoid snapshot costs for no-backup tables, truncate them before taking a snapshot.

A backup, or *recovery point*, represents the content of a resource, such as an Amazon Redshift cluster, at a specified time. AWS Backup saves backups in backup vaults, which you can organize according to your business needs. The terms *recovery point* and *backup* are used interchangeably. For more information about AWS Backup, see Backup creation, maintenance, and restore in the *AWS Backup Developer Guide*.

Amazon Redshift is natively integrated with AWS Backup. That lets you define your backup plans and assign Amazon Redshift resources to the backup plans. AWS Backup automates the creation of Amazon Redshift manual snapshots, and securely stores these snapshots in a backup vault that you designate in your backup plan. For information about vaults, see Backup vaults in the *AWS Backup Developer Guide*. In the backup plan, you can define the backup frequency, backup window, lifecycle, or backup vault. For information about backup plans, see Backup plans in the *AWS Backup Developer Guide*.

For information about creating and restoring Amazon Redshift Serverless snapshots without using AWS Backup, see Snapshots and recovery points. For information about creating and restoring Amazon Redshift provisioned cluster snapshots without using AWS Backup see Amazon Redshift snapshots and backups.

Topics

- Considerations for using AWS Backup with Amazon Redshift
- Limitations for using AWS Backup with Amazon Redshift
- Managing AWS Backup with Amazon Redshift

Considerations for using AWS Backup with Amazon Redshift

Following are considerations for using AWS Backup with Amazon Redshift:

- AWS Backup for Amazon Redshift is available where both AWS Backup and Amazon Redshift are available in the same AWS Regions. For information on where AWS Backup is available, see Amazon Redshift endpoints and quotas in *AWS General Reference*.
- To get started using AWS Backup, verify that you have met all the prerequisites. For more information, see Prerequisites in the *AWS Backup Developer Guide*.
- Affirmatively opt in to AWS Backup service. Opt-in choices apply to the specific account and AWS Region. If you want to use backups in multiple Regions with a given account, you must opt in to each individual Region with that account. For more information, see Opt in to managing services with AWS Backup in the *AWS Backup Developer Guide*.
- AWS Backup integration for Amazon Redshift only supports manual snapshots for provisioned clusters and serverless namespaces.
- Once you use AWS Backup to manage snapshot settings, you can't continue to manage manual snapshot settings using Amazon Redshift. Instead, you can continue to manage the settings using an AWS Backup plan. For more information, see Backup plans in the *AWS Backup Developer Guide*.
- Restoring whole data warehouse snapshots to a serverless namespace is a destructive change. All previously existing data in the target namespace is lost when you restore a data warehouse snapshot to that namespace. This only applies to restoring data warehouse snapshots. Restoring single table snapshots to a namespace doesn't delete existing data.
- To restore a snapshot to a provisioned cluster, you need to have an IAM policy with the `RestoreFromClusterSnapshot` permission. To restore a snapshot to a serverless namespace, you need to have an IAM policy with the `RestoreFromSnapshot` permission. These permissions apply to the target data warehouse type, not to source snapshot type. For example, to restore a cluster snapshot to a namespace, you would need the `RestoreFromSnapshot` permission, not `RestoreFromClusterSnapshot`. For more information on managing IAM policies, see Identity and access management in Amazon Redshift.

Limitations for using AWS Backup with Amazon Redshift

Following are limitations for using AWS Backup with Amazon Redshift:

- You can't use AWS Backup to manage Amazon Redshift automated snapshots. To manage automated snapshots, use tags. For information about tagging resources, see Tagging resources in Amazon Redshift.
- When restoring single tables from a snapshot, you can’t restore from a provisioned cluster snapshot to a serverless namespace or vice versa. You can restore entire snapshots in any configuration. For example, you can restore all of the databases in a provisioned cluster snapshot to a serverless namespace, but you can’t restore a single table from that same snapshot to the same namespace.

Managing AWS Backup with Amazon Redshift

To protect resources on your Amazon Redshift data warehouses, you can use the AWS Backup console, or programmatically use the AWS Backup API or AWS Command Line Interface (AWS CLI). When you need to recover a resource, you can use either the AWS Backup console or the AWS CLI to find and recover the resource you need. For more information, see AWS Command Line Interface.

When using AWS Backup for Amazon Redshift, you can perform the following actions:

- Create periodic backups that automatically initiate Amazon Redshift snapshots. Periodic backups are useful to meet your long-term data retention needs. For more information, see Amazon Redshift backups in the *AWS Backup Developer Guide*.
- Automate backup scheduling and retention by centrally configuring backup plans.
- Restore a provisioned cluster or serverless namespace to the saved backup you choose. You can choose to restore all of the data in the snapshot or a single table from it. You set how often to back up your resources. For information about restoring provisioned cluster snapshots, see Restore an Amazon Redshift cluster in the *AWS Backup Developer Guide*. For information on restoring serverless namespace snapshots, see Amazon Redshift Serverless restore in the *AWS Backup Developer Guide*.

Cluster versions for Amazon Redshift

Amazon Redshift regularly releases cluster versions. Your Amazon Redshift clusters are patched during your system maintenance window. The timing of the patch depends on your AWS Region and maintenance window settings. You can view or change your maintenance window settings from the Amazon Redshift console. For more information about maintenance, see Cluster maintenance.

You can view the cluster version of your cluster on the Amazon Redshift console on the **Maintenance** tab of the cluster details. Or you can see the cluster version in the output of the SQL command:

```
SELECT version();
```

> **Note**
>
> Critical updates that affect Amazon Redshift behavior are introduced as Amazon Redshift evolves. To keep up with these changes, take actions, and avoid potential disruptions to your workloads, see Behavior changes in Amazon Redshift.

Topics

- Amazon Redshift patch 206
- Amazon Redshift patch 205
- Amazon Redshift patch 204
- Amazon Redshift patch 203
- Amazon Redshift patch 202
- Amazon Redshift patch 201
- Amazon Redshift patch 200
- Amazon Redshift patch 199
- Amazon Redshift patch 198
- Amazon Redshift patch 197
- Amazon Redshift patch 196

- Amazon Redshift patch 195
- Amazon Redshift patch 194
- Amazon Redshift patch 193
- Amazon Redshift patch 192
- Amazon Redshift patch 191
- Amazon Redshift patch 190
- Amazon Redshift patch 189
- Amazon Redshift patch 188
- Amazon Redshift patch 187
- Amazon Redshift patch 186
- Amazon Redshift patch 185
- Amazon Redshift patch 184
- Amazon Redshift patch 183
- Amazon Redshift patch 182
- Amazon Redshift patch 181
- Amazon Redshift patch 180
- Amazon Redshift patch 179
- Amazon Redshift patch 178
- Amazon Redshift patch 177
- Amazon Redshift patch 176
- Amazon Redshift patch 175
- Amazon Redshift patch 174
- Amazon Redshift patch 173
- Amazon Redshift patch 172
- Amazon Redshift patch 171
- Amazon Redshift patch 170
- Amazon Redshift patch 169
- Amazon Redshift patch 168

Amazon Redshift patch 206

Cluster versions in this patch:

- 1.0.520420 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released October 27, 2026

  1.0.485645 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released October 13, 2026

New features and improvements in this patch

- Added the `PG_LAST_QUERY_UUID()` function, which returns the UUID of the last completed query.
- Added a `session_id` column to `STL_UNLOAD_LOG` that records the leader node's session identifier that matches the `session_id` value in `SYS_QUERY_HISTORY`.
- Added an `external_user_id` column to STL and STV system tables and SYS monitoring views for consistent identity tracking across provisioned clusters and serverless workgroups.
- Added opt-in support for storing string values up to 16,000,000 bytes in the `SUPER` data type.
- Added support for Amazon Redshift Federated Permissions in multi-warehouse configurations.
- Added support for sharing stored procedures through Amazon Redshift data sharing. Consumers can now call stored procedures shared by a producer and run them on the consumer's own compute.
- Improved cluster stability by reducing node restarts that could occur while committing transactions under high memory pressure.
- Fixed an issue where a database name with leading or trailing spaces could cause a system table upgrade to fail and the cluster to enter a restart loop.
- Fixed an issue where queries with large parameters could cause a session to terminate unexpectedly.
- Improved the reliability of streaming ingestion from Amazon Kinesis Data Streams into streaming materialized views with auto refresh, preventing unexpected interruptions and restarts.
- Improved zero-ETL integration recovery after cluster restarts by skipping unnecessary catalog updates when the integration database state is unchanged.

- Improved zero-ETL ingestion throughput on larger clusters (4xlarge-class node types and above) by allowing more concurrent ingestions per database when there is sustained backlog.
- Auto refresh interval for materialized views (MVs) -- We are introducing a new capability to set a user configurable MV auto refresh interval. This aligns MV auto refresh behavior with ETL (extract transform load) SLAs (service level agreements) by providing you an option to set when the MV auto refresh is triggered.
- Vacuum of tables with dependent MVs -- With this enhancement, the refresh of materialized views (MVs) defined on local RMS (Redshift Managed Storage) tables stay incremental after a vacuum operation is run on the MV's base tables.
- Materialized View internal backing tables are now stored in an internal protected schema.
- AQMV support for MVs on Iceberg tables
- Changed how tables load after a cluster restart so that commits no longer wait for all tables to load. The first commit can complete up to 15 minutes sooner, with minimal impact on query performance. This improvement is currently enabled for a limited set of Amazon Redshift Serverless workgroups and will be extended in future patches.
- Fixed a rare issue where a disk error during internal cluster maintenance could cause an unexpected node restart.
- Fixed a rare issue where memory pressure could cause an unexpected node restart.
- Fixed an issue where files belonging to dropped databases were not cleaned up promptly under heavy `DROP DATABASE` activity.
- Fixed an issue where query admission on clusters using automatic workload management could cause an unexpected cluster restart.
- Fixed an issue where queries that reference objects in another Amazon Redshift warehouse could slow down when that warehouse contains a large number of tables.
- Fixed an issue where a Lambda user-defined function in a join condition could cause an unexpected query failure when it referenced neither joined table.
- Fixed a query failure that could occur when `ST_Collect` was combined with a `DISTINCT` aggregate (such as `COUNT(DISTINCT)`) in the same grouped aggregation
- Fixed a query failure on an external (Spectrum/Iceberg) table that combined a Lambda UDF with a hash join and dynamic partition elimination
- Provided a fix to improve reliability of online resize operations. A failure where the transition to read-only mode could stall and never complete is fixed, so these operations no longer get stuck partway through and complete successfully without manual intervention.

- Fixed an issue where queries that call AWS Lambda user-defined functions (UDFs) could exhaust internal process resources and cause an unexpected cluster restart.
- Improved error messages for zero-ETL integrations to provide clearer guidance on recovery behavior, including when Amazon Redshift automatically resynchronizes affected tables.

Amazon Redshift patch 205

Cluster versions in this patch:

- 1.0.501809 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released October 19, 2026
- 1.0.477953 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released October 6, 2026
- 1.0.434008 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released September 16, 2026

New features and improvements in this patch

- Iceberg MV – Materialized views stored in Iceberg offering incremental refresh and full recompute.
- System tables now include an `external_user_id` column that records the durable IAM or IAM Identity Center identity associated with each query, enabling per-user filtering across cluster lifecycle events.
- Added support for the IAM_ROLE 'SESSION' keyword for COPY and UNLOAD, allowing customers to use session-bound FAS credentials for Amazon S3 access instead of specifying an explicit IAM role ARN.
- Added `ALTER DATABASE ... INTEGRATION REFRESH REMEDIABLE TABLES`, which resynchronizes zero-ETL tables flagged with a duplicate-date warning on demand, optionally scoped to a schema.
- Added the `IntegrationLatestSourceCommit` Amazon CloudWatch metric for zero-ETL integrations, reporting the latest source commit time observed so you can distinguish an idle source from a stalled pipeline.
- Amazon Redshift now lets an administrator raise the minimum length required for database user passwords using `ALTER SYSTEM SET passwd_min_length = N` (8–32).

- Fixed an issue where `ALTER IDENTITY PROVIDER ... ENABLE` failed with an incorrect-parameters error on Azure identity providers configured to automatically create roles.
- Amazon Redshift now supports post-quantum hybrid key exchange (`X25519MLKEM768`) for TLS connections.
- Amazon Redshift now supports a `DEBUG` permission on federated permissions catalog databases, letting a global identity, an IAM Identity Center (IdC) group, or a consumer account's administrators view unredacted secure logging records for fine-grained access control (FGAC)-protected data.
- Amazon Redshift now supports scoped permissions with Metadata Security.
- Iceberg tables can now be created with a NOT NULL column constraint.
- Amazon Redshift now supports IANA Time Zone Database version 2026c.
- Amazon CloudTrail audit events for Amazon Redshift-initiated API calls now include a `queryUuid` field for correlation with system tables.
- Renamed the `user_query_uuid` column to `query_uuid` in system tables and views for consistency.
- Improved query performance and availability on clusters with zero-ETL integrations under high query concurrency.
- Permission checks on tables with a large number of columns and few column-level permissions are now faster, which speeds up metadata queries and client tools that list table privileges.
- Enhanced stability for queries that use Lambda UDFs in Aggregates.
- Automatic bloat removal now lets the superblock shrink to fit its live header set, removing the need to reclaim bloat through Superblock Vacuum (SBV).
- Improved automatic table sort to reduce storage growth on tables with a large number of columns.
- Fixed an issue that could stop automatic vacuum from running on a cluster until the cluster was restarted.
- Fixed an issue that could cause a cluster restart during a vacuum operation on clusters using automatic workload management with concurrency scaling.
- Fixed a bug that could lead to errors with type inference for recursive CTEs containing string literals.
- Fixed an issue where federated PostgreSQL queries could hang on an unresponsive remote database and ignore cancellation requests.

- Fixed a rare cluster restart caused by a double-free in COPY load state management under memory pressure.
- Fixed an issue where `SHOW` discovery commands could not find results when given a long pattern in the `LIKE` clause.
- Fixed an issue where creating or publishing a datashare could fail on a cluster that had reached its maximum number of database roles.
- Canceling a `DROP DATABASE` command now ends the command with an error instead of causing an unexpected cluster restart.
- Improved reliability of stored procedures that query Iceberg tables multiple times followed by a query on a local table when concurrency scaling is in use.
- Improved reliability of sessions that mix Iceberg cursors with non-Iceberg queries when concurrency scaling is enabled.
- Added the capability to make the fdisk fetch-block IPC wait cancellable so a lost ACK becomes a killable query instead of an unkillable hang.
- Improved the reliability of the internal process used during Online Elastic Resize to make the process more deterministic/robust.

Amazon Redshift patch 204

Cluster versions in this patch:

- 1.0.436211 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released September 22, 2026
- 1.0.436211 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released September 10, 2026
- 1.0.416217 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released August 25, 2026
- 1.0.394035 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released August 11, 2026

New features and improvements in this patch

- Added support for single-node rg.large clusters. This is the minimum required cluster version to create a single-node rg.large cluster.

- Added a query_uuid column to system tables and views, providing a globally unique query identifier that persists across cluster pause/resume cycles.
- Added user_query_id to stl_udf_log table to remove dependency on stl_user_query_map joins in sys_ view definitions.
- Added two new Amazon CloudWatch metrics, IntegrationLatestDetectedChange and IntegrationLatestAppliedChange, for zero-ETL integrations to allow monitoring of replication health.
- Added the AUTO_REMEDIATION option for CREATE DATABASE and ALTER DATABASE on zero-ETL integration databases. When AUTO_REMEDIATION is set, Amazon Redshift automatically resynchronizes tables that have replication issues. Details appear in svv_integration_table_state.reason.
- Added support for reading, creating, and writing to Apache Iceberg v3 tables using the Amazon Redshift native engine. This includes support for default values, row lineage, and deletion vectors across all previously supported data types. Support for the Variant, Geography, and Geometry data types is not included in this release and will be announced in a future patch.
- After upgrading an Iceberg table to v3, the Iceberg `timestamptz` type is mapped to the Amazon Redshift `TIMESTAMPTZ` type. In Iceberg v2 tables, `timestamptz` is mapped to the Amazon Redshift `TIMESTAMP` type. With v3, this means your queries output the timestamp based on their timezone.
- Added support for `CURSOR`s that exceed the maximum result set size for the instance type. When a cursor's result set exceeds the available disk capacity, Amazon Redshift automatically spills overflow data to Amazon S3. This feature can be enabled per cluster upon request to AWS Support.
- Added new bin packing metrics columns to the `stl_fasma_data_cache_stats` system table for monitoring cache activity.
- Parsing and semantic analysis of queries that reference Late Binding Views (LBVs) now occurs on consumer clusters. This removes a producer-side bottleneck in LBV processing.
- Amazon Redshift ML now enables the inter-container traffic encryption option when it makes `CreateAutoMLJob` and `CreateProcessingJob` requests to Amazon SageMaker. This aligns with the setting already applied to `BuildTrainingRequest` and prevents security scanner alerts.
- Improved zero-ETL integration performance for high-frequency workloads to reduce replication lag for tables with many concurrent updates.

- Zero-ETL integrations now correctly report source data exceeding the target column length as a non-retriable error.
- Improved zero-ETL integration stability after cluster restarts.
- Empty schemas are now automatically cleaned up when a table filter change on a zero-ETL integration removes all tables from a schema.
- Improved the error message when attempting DROP DATABASE on a database with an active zero-ETL integration.
- DynamoDB zero-ETL integrations now honor the ACCEPTINVCHARS and TRUNCATECOLUMNS options during both initial seeding and ongoing replication. Tables continue to replicate when the source contains invalid UTF-8 characters.
- Improved query performance on clusters with zero-ETL integrations, especially for workloads that replicate many databases or tables.
- Improved performance for workloads using temporary tables, reducing catalog access overhead and improving overall throughput.
- SHOW discovery commands such as SHOW TABLES, SHOW SCHEMAS, SHOW COLUMNS, and SHOW GRANTS now return up to 133,500 objects, so metadata listings on schemas and databases with very large object counts succeed instead of failing with a row-limit error of 10,000.
- When a DROP USER command is rejected because the user still owns objects or holds privileges, the error message now names the specific blocking object and reports how many other dependencies remain.
- Amazon Redshift now caches IAM role credentials on the leader node. This reduces retrieval overhead for COPY, UNLOAD, Amazon Redshift Spectrum, and Lambda UDF queries. Frequently used credentials are reused across queries, improving query start-up time.
- AWS IAM Identity Center authentication now works on clusters and workgroups with enhanced VPC routing enabled.
- Improved disk-health detection for RG node types to prevent unnecessary automatic hardware remediations on clusters operating under normal disk I/O.
- Enhanced Data Sharing performance by skipping an extra copy in the backend process.
- Expanded consumer size support for multi-warehouse write queries.
- Enhanced stability for Lambda User-Defined Functions with no arguments in joins.
- Fixed an issue in MDDL sort key computation when predicates include IDF.

- Fixed an issue where queries on external tables, data sharing objects, and cross-database references failed with "transaction is read-only" when executed inside BEGIN READ ONLY transactions.
- Fixed a rare condition where a COPY or INSERT running at the same time as automatic table optimization on the same table could cause a cluster restart.
- Fixed a rare encoding-mismatch error that could cause automatic table optimization to fail on tables accessed through data sharing that were created on older cluster versions.
- Fixed a rare condition where automatic vacuum on tables with skewed data distribution could cause a cluster restart.
- Fixed an issue where an Auto COPY job could stop making progress and block other Auto COPY operations on the same cluster.
- Fixed a rare condition where a COPY command could cause a cluster restart due to a race condition during query cleanup.
- Fixed an issue where svl_s3query_summary.s3query_returned_rows could report one extra row for queries with a LIMIT clause on external tables.
- Fixed a rare cluster restart that could occur during bootstrap on clusters with zero-ETL integrations and no Auto COPY jobs configured.
- Fixed an issue with COPY field parsing when using a multi-character delimiter with the ESCAPE option.
- Fixed an issue where COPY and UNLOAD commands failed when users connected via IAM Identity Center provided explicit temporary credentials using the CREDENTIALS parameter.
- Fixed an issue in SHOW CONSTRAINTS that could cause cluster restarts.
- Fixed an issue where CONNECT permission granted to a role did not allow AWS IAM Identity Center users in the mapped group to connect on databases with Amazon Redshift federated permissions enabled.
- Fixed an issue where queries using UNNEST on SUPER arrays were incorrectly blocked from running on concurrency scaling clusters.
- Fixed an issue where the ARRAY_SORT function could fail with a path navigation error when applied to nested SUPER structures.
- Fixed an issue where deeply nested PartiQL or SUPER expressions could cause a cluster restart.
- Fixed an issue where the TO_NUMBER function could fail when the format string specified a precision greater than 38.

- Fixed an issue where encryption key rotation could corrupt keys for grafted tables, potentially causing data access failures.
- Fixed deadlock conditions in storage operations that could cause query hangs or node unresponsiveness.
- Fixed a rare crash during disk block management that could cause node restarts.
- Fixed a crash caused by out-of-memory during block allocation that could cause node restarts.
- Fixed a crash on disk-full conditions during block replication assignment that could cause node restarts.

Amazon Redshift patch 203

Cluster versions in this patch:

- 1.0.417950 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released August 26, 2026
- 1.0.417950 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released August 25, 2026
- 1.0.400034 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released August 17, 2026
- 1.0.384821 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released August 4, 2026
- 1.0.377293 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released July 25, 2026
- 1.0.365190 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released July 21, 2026
- 1.0.347559 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released July 6, 2026

New features and improvements in this patch

- A new column was added to stl_alter_dist_sort to indicate whether ALTER DISTKEY and ALTER SORTKEY operations executed on the main cluster, a data sharing consumer, or a concurrency scaling cluster.

- Added SHOW GRANTS FOR GROUP syntax to discover permissions granted to a specific user group.
- Enhanced zero-ETL integration observability by adding granular success status for checkpoints, enabling finer-grained monitoring of replication progress in system tables.
- Introduced graceful handling of unsupported UTF-8 characters in source table schemas and DDL metadata within zero-ETL integrations, preventing integration failures.
- Amazon Redshift now supports concurrency scaling for refreshes of Amazon Kinesis Data Streams (KDS)-connected streaming materialized views (MV).
- Starting with Patch 203, Amazon Redshift begins enforcing the end of support for existing Python UDFs. Creation of new Python UDFs was deprecated in Patch 198, and all Python UDF support ends after June 30, 2026. Migrate your existing Python UDFs to Lambda UDFs to ensure continued functionality. For migration guidance, see Amazon Redshift Python user-defined functions will reach end of support after June 30, 2026, and Best practices for Amazon Redshift Lambda user-defined functions on the AWS Big Data Blog. For details on behavioral changes, see Scalar Python UDFs will reach end of support after June 30, 2026.
- Tables loaded with COPY from Amazon Redshift Spectrum, with explicit column encodings, or into non-empty tables are now eligible for automatic DISTKEY assignment when the table has a primary key, improving query performance for joins on the primary key.
- Updated the ALTER TABLE … ALTER DISTKEY error message on data sharing tables to display the customer-facing table name.
- Reduced background activity from automatic vacuum on clusters with a very large number of tables, improving cluster stability for clusters that previously experienced excessive maintenance activity.
- Optimized query performance by combining subqueries with mismatched filters, aggregation, and window functions in certain cases.
- Improved timestamp accuracy for time-travel queries on zero-ETL tables with Amazon DynamoDB history mode following a table resynchronization.
- Zero-ETL integrations no longer enter a resync loop when processing CDC checkpoints on tables with certain primary key configurations.
- Improved zero-ETL integration error reporting for invalid channel filter expressions.
- Original query text is no longer masked with Secure Logging (applies to queries against external AWS Lake Formation view or database with Federated Amazon Redshift Permissions).
- Improved the performance of ANALYZE for large tables by up to 30%.

- Enhanced performance for queries using AWS Lambda User-Defined Functions with ROLLUP aggregations.
- Enhanced query performance for ORDER BY with LIMIT clause by skipping unnecessary data scans on the outer table of Hash Join.
- Reduced memory consumption during background tree-sweep operations eliminating node crash fleet wide. Bounded the memory footprint of compute-node tree-sweep by implementing survivor-counted batch drain, preventing unbounded memory growth under high metadata churn. Tree-sweep paths are now gated on background tree replay completion.
- Improved fault recovery after Amazon S3 read failures, eliminating node crashes. The trickle tree iterator is now properly reset after an Amazon S3 read failure, ensuring subsequent reads retry correctly instead of silently returning stale or incomplete data.
- Fixed an issue where tables that were modified with ALTER TABLE … ALTER DISTSTYLE AUTO after an Adaptive Cluster Resize could remain in Pending state in sys_restore_state indefinitely.
- Fixed an issue where the SYS_QUERY_TEXT system view could display query text with missing whitespace or incorrectly split words at chunk boundaries.
- Fixed a startup failure because of zero-ETL integrations for clusters created between August 2023 and March 2024, caused by Amazon Redshift Catalog metadata inconsistencies.
- Fixed an issue where SHOW GRANTS incorrectly displayed permissions granted to a user group with identity_type "role".
- Fixed an issue where queries could hang indefinitely during session suspension handling.
- Fixed an issue where non-superusers with SYSLOG ACCESS UNRESTRICTED could only see their own session data in sys_session_history instead of all users' sessions as expected.
- Fixed an issue where database or user connection limits set to 1000 or higher were truncated, preventing the correct number of connections.
- Fixed an issue where queries could hang indefinitely under memory pressure.
- Fixed an issue where Amazon Redshift Serverless endpoints could experience brief unavailability during pause operations.
- Fixed an issue where COPY commands with certain parameter combinations could fail unexpectedly during query compilation.
- Fixed an issue where queries using APPROXIMATE PERCENTILE_DISC with GROUP BY could fail with a compilation error.
- Fixed a wrong-results issue where queries using COLLATE expressions could return incorrect column values.

- Fixed an issue where LEAST and GREATEST functions with SUPER type values could cause queries to fail.
- Fixed a wrong-results issue in REGEXP_REPLACE where replacement strings with capture group references could produce truncated output.
- Fixed potential data corruption during aborted re-replication reads. Resolved a double-free condition that could occur when a re-replication read was aborted mid-flight, preventing possible memory corruption or crashes.

Amazon Redshift patch 202

Cluster versions in this patch:

- 1.0.369760 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released July 29, 2026
- 1.0.377292 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released July 25, 2026
- 1.0.377292 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released July 25, 2026
- 1.0.358853 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released July 21, 2026
- 1.0.358853 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released July 14, 2026
- 1.0.343555 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released June 29, 2026
- 1.0.340728 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released June 22, 2026
- 1.0.331549 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released June 16, 2026
- 1.0.323213 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released June 3, 2026
- 1.0.314775 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released May 28, 2026

New features and improvements in this patch

- Added support for RG node types (rg.large and rg.12xlarge multi-node). This is the minimum required cluster version to migrate to rg.large and rg.12xlarge.
- Added support for superusers to drop IAM Identity Center identity providers that are in a disabled state of Lakehouse type
- Surfaced internal maintenance queries labeled "JIT ANALYZE" in the STL_QUERY and STL_QUERYTEXT system tables. These queries are automatically executed by Amazon Redshift to collect external table statistics and keep query plans optimized. No customer action is required.
- We have optimized Kinesis stream identity validation to reduce `DescribeStreamSummary` API calls. This will avoid hitting Kinesis API quotas for Redshift customers with many Kinesis stream MVs. See the Redshift Streaming documentation section for details.
- Enhanced query execution performance by using Top K Aggregation for queries with GROUP BY, ORDER BY, and LIMIT clauses, reducing the amount of data processed during aggregations.
- Enhanced performance for queries on partitioned AWS Glue Data Catalog tables by pushing compound filter predicates (OR, IN, NOT IN) to the Glue GetPartitions API, reducing the number of partitions retrieved.
- Enhanced performance for queries using Lambda User-Defined Functions with Window functions.
- Enhanced performance for particular queries using Lambda User-Defined functions in Join Filters of Hash Joins.
- Enhanced stability for queries that use Lambda UDFs in Joins.
- Allow using ARN as the name of AWS Lambda when creating Lambda User-Defined functions.
- Improved query performance by optimizing memory usage for COALESCE and CASE WHEN expressions.
- Applies advanced filtering to queries containing subplans that are evaluated once, with their results reused as constants throughout the remaining execution.
- Improved performance of the first time run queries using ST_Transform
- Improved Apache Iceberg compatibility by accepting positional delete files that contain duplicate row indices, which can be produced by AWS Glue compaction.
- Improved Apache Iceberg compatibility by accepting s3a:// paths in delete file references.
- Improved handling of the Spectrum partition level fetching errors when processing not properly configured tables

- Improved stability of stored procedure that invokes nested SQL UDFs across databases via USE
- Improved error messages for unsupported operations on stored procedures, such as adding or removing them from datashares.
- Fixed a bug that prevented detection of customer's VPC information when EVR is in use. This information is only necessary for training local Redshift ML models, and has no impact on any other Redshift functionality. This did not prevent the use of existing trained Redshift ML models.
- Fixed an issue where duplicate query IDs could appear in system tables after a cluster pause and resume when the internal query ID counter was near its maximum value.
- Fixed an issue where queries on STL system tables could return incomplete results after a Multi-AZ failover due to an internal caching inconsistency
- Fixed an issue where the "isAutoIncremented" field in extended row descriptions could return incorrect values for IDENTITY columns
- Fixed an issue where the ARRAY_CONTAINS and ARRAY_POSITION functions did not return NULL as expected when given NULL inputs.
- Fixed a rare issue where queries with multiple combined filter predicates during vectorized scans could fail unexpectedly.
- Fixed a rare issue where concurrent COPY operations could cause cluster instability in the data loading process
- Fixed an issue where automatic maintenance operations such as vacuum and analyze could cause cluster instability during process shutdown
- Fixed an issue where queries could fail after workload management (WLM) short query acceleration was dynamically toggled off and back on
- Fixed an issue where time-travel queries on zero-ETL tables with history mode.
- Fixed an issue where zero-ETL integrations could become unresponsive during concurrent client shutdown
- Fixed an issue where zero-ETL tables sharing a source shard could trigger unnecessary resynchronization
- Fixed an issue where zero-ETL integration table state updates could fail when the internal reason field exceeded the allowed length
- Fixed an issue where zero-ETL integrations encountered errors during column changes on an Aurora PostgreSQL partitioned table

Amazon Redshift patch 201

Cluster versions in this patch:

- 1.0.331388 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released June 16, 2026
- 1.0.331388 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released June 15, 2026
- 1.0.326225 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released June 8, 2026
- 1.0.319797 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released June 3, 2026
- 1.0.313685 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released May 27, 2026
- 1.0.300094 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released May 12, 2026
- 1.0.298697 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released May 7, 2026
- 1.0.292316 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released May 4, 2026
- 1.0.280974 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released April 22, 2026

New features and improvements in this patch

- Added support for RG node types (rg.xlarge and rg.4xlarge multi-node). This is the minimum required cluster version to migrate to rg.xlarge and rg.4xlarge.
- Added support for ALTERs on Iceberg tables.
- Added support to write to Iceberg tables mounted under AwsDataCatalog.
- Added support for the SHOW GRANTS command when connected to a database using the USE command.
- Added support for SHOW GRANTS and SHOW COLUMN GRANTS commands for identities with federated permissions.

- Added support for materialized view auto-refresh as a user workload with burst capacity on Amazon Redshift Serverless.
- Allow AutoWLM to assign more cluster memory to queries writing to wide tables.
- Amazon Redshift improved the query performance of the SVV_ATTACHED_MASKING_POLICY system view, reducing execution time on instances with a large number of permissions and dynamic data masking policy attachments.
- Upgraded the librdkafka library to version 2.13.x from 2.8.x to provide the latest bug fixes and performance enhancements.
- Enhanced performance for queries using Lambda User-Defined Functions in the HAVING clause of aggregates.
- Enhanced performance for queries using Lambda User-Defined Functions with window functions.
- Enhanced stability for Lambda User-Defined Functions with no arguments.
- Enhanced datalake table query execution performance when using ORDER BY and LIMIT clauses by eliminating unnecessary data scans.
- Improved error messages for zero-ETL integrations to provide clearer details for unsupported constraints, unsupported DDL statements, date precision loss, multibyte character length mismatches, and skipped source tables.
- Improved zero-ETL integration resilience by gracefully handling credential acquisition failures during replication status checks.
- Improved performance of zero-ETL integrations when running concurrent replication tasks.
- Improved time-travel query results for zero-ETL tables with frequently updated rows to ensure each historical row version reflects its individual deletion timestamp.
- Fixed some edge cases where the main cluster was not properly resized by AI-driven scaling.
- Fixed an issue where Amazon CloudWatch metrics could report duplicate failed query counts when queries were evicted and retried by workload management (WLM).
- Fixed an issue where COPY with Redshift Table Copy could produce a distribution style mismatch.
- Fixed an issue with incorrect partition sort end row calculation that could affect range-restricted scan results.
- Fixed an issue where automatic vacuum operations could fail when a table was dropped concurrently.
- Fixed an issue where VACUUM could use a stale table version when acquiring the relation, potentially leading to errors.

- Ensured vacuum transaction map is properly cleaned up on burst clusters.
- Fixed an issue where queries on zero-ETL replicated tables could fail when accessed through data sharing.
- Fixed an issue in zero-ETL integration data replication that could cause cluster restarts.
- Fixed an issue where zero-ETL integrations with DynamoDB history mode could fail during update-merge operations.
- Fixed an issue where zero-ETL integration tables could display incorrect row counts in the Amazon Redshift console.
- Fixed an issue that prevented Amazon Redshift Serverless workgroups from pausing during zero-ETL integration operations.
- Fixed an issue where empty Avro files caused COPY ingestion to fail. Empty Avro files are now skipped and ingestion continues without an error.
- Fixed a stale block header checksum issue during epsilon tree replay when using position-independent superblock.

Amazon Redshift patch 200

Cluster versions in this patch:

- 1.0.314426 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on May 27, 2026
- 1.0.293472 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on May 7, 2026
- 1.0.293472 – **CURRENT Tracks** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on April 30, 2026
- 1.0.266483 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on April 9, 2026
- 1.0.257586 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on March 27, 2026

New features and improvements in this patch

- Fixed an issue where INSERT operations could fail on tables using multidimensional data layout sort keys.

- Improved Amazon Redshift Advisor recommendations for multidimensional data layout (MDDL) sort keys for tables that would benefit from workload-based sorting.
- Enhanced Redshift Serverless AI-driven scaling and optimizations with improved disk usage signals, enabling more accurate and responsive scaling decisions.
- Enhanced Redshift Serverless AI-driven scaling and optimizations to intelligently determine an initial capacity when you migrate a provisioned cluster snapshot to a serverless namespace.
- Fixed an issue where data sharing queries using IAM Identity Center identity propagation could fail when the cached identity token expired on the consumer.
- Fixed an issue where IAM Identity Center group synchronization could cause subsequent background maintenance operations to fail.
- Fixed an issue where updating an IAM Identity Center identity provider configuration with a different application ARN could cause a cluster restart.
- Fixed an issue where data sharing queries with IAM Identity Center identity propagation could fail in certain AWS Regions.
- Improved IAM Identity Center integration reliability by ensuring identity provider configuration changes are consistently propagated.
- Optimized query plans for some correlated NOT EXISTS subqueries by decorrelating them into anti-joins.
- Fixed an issue where DROP TABLE IF EXISTS on Apache Iceberg tables could incorrectly report a query failure instead of success when the target table did not exist.
- Added proper error messaging when users attempt to run ANALYZE on catalog tables.
- Spectrum COPY is now generally available, removing the dependency on temporary tables for Spectrum-based COPY operations.
- Improved Auto COPY reliability by throttling S3 notification queue reads when durable persistence tasks cannot keep pace, reducing the need for S3 listing operations to reconcile COPY JOB state.
- Concurrency Scaling is now generally available for read and write operations on zero-ETL tables.
- Fixed a race condition in zero-ETL CDC replication that could cause cluster restarts when concurrent disk commits interfered with uncommitted table state tracking.
- Improved zero-ETL CDC ingestion reliability by using the most recent ingestion position across both table and client state to filter already-applied data.

- Fixed a data reliability issue in zero-ETL table resync where old, already-applied checkpoints could be re-ingested after table failures, by correctly tracking checkpoint lower bounds and resync state across the re-synchronization lifecycle.
- Improved zero-ETL error messages with clearer failure reasons, including details for column length exceeded and invalid UTF-8 character errors.
- Enhanced the SYS_LUDF_DETAIL system view to report duration in microseconds instead of milliseconds for improved observability.
- Fixed an issue where Auto COPY job creation was incorrectly allowed in zero-ETL databases.
- Improved query performance on row-level security (RLS) protected tables by classifying additional built-in functions as RLS-safe, enabling predicate pushdown for common query patterns that previously experienced performance degradation.
- Added support for DELETE, UPDATE, and MERGE operations on Apache Iceberg tables.
- Added support for position-independent superblock, enabling predictable cluster restart times regardless of data volume.
- Enabled GUID-based disk cache, removing the dependency of disk cache on superblock position and enabling predictable cluster restart times regardless of data volume.
- Improved automatic table distribution optimization: tables with a single primary key are now eligible for automatic conversion from ALL distribution to KEY distribution, improving query performance and reducing storage overhead.
- Fixed an issue where VACUUM operations could fail or produce incorrect results due to a conflict when vacuum and concurrent DML operations accessed the same storage object.
- Fixed an issue where VACUUM FULL could fail on certain tables due to an out-of-bounds memory access when processing special partition chains.
- Fixed an issue where ALTER TABLE OWNER operations could incorrectly allow changing object ownership to the internal bootstrap user.
- Fixed a race condition where concurrent VACUUM operations and staging table reuse could cause cluster instability.
- Fixed an issue where VACUUM FULL or VACUUM SORT on large tables could fail with an internal error when the number of processed partitions exceeded an internal limit.
- Fixed an issue where background partition statistics collection could fail when a table was dropped while statistics were being aggregated from remote clusters.
- Improved performance of VACUUM SORT operations by avoiding unnecessary data refreshes, reducing the time and resources required for vacuum on sorted tables.

- Fixed an issue where incorrect internal partition data access during vacuum operations could cause unexpected errors or cluster instability.
- Fixed an issue where ALTER TABLE operations could incorrectly change table ownership to a non-privileged user. Owner changes during ALTER operations are now correctly restricted to authorized users only.
- Fixed a rare issue where an internal limit check during vacuum operations could cause unnecessary query failures on tables with a large number of partitions.
- Fixed an issue where background statistics collection could fail or produce errors when encountering tables that had been deleted, improving cluster stability.
- Fixed an issue where vacuum operations could use stale table metadata, potentially causing incorrect behavior during vacuum on tables with concurrent schema changes.
- Released a new query compilation enhancement that increases performance for new queries (first runs of queries) in Amazon Redshift patch P198 on the CURRENT Maintenance Track. Now, this feature is enabled on all releases starting with Amazon Redshift patch P200 on the TRAILING track as well.

Amazon Redshift patch 199

Cluster versions in this patch:

- 1.0.266409 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on April 20, 2026
- 1.0.252420 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on March 25, 2026
- 1.0.232773 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on March 5, 2026

New features and improvements in this patch

- Released a new query compilation enhancement that increases performance for new queries (first runs of queries) in the Redshift CURRENT Maintenance Track. The TRAILING track will follow in an upcoming patch release.
- Enabled support for Column level grants from a consumer using Redshift federated permissions for a new grantee on producer.

- Amazon Redshift federated permissions supports querying external tables through Late Binding Views (LBVs).
- ALTER EXTERNAL SCHEMA can now be used with the KINESIS external schema type to change the IAM_ROLE and REGION of an existing external schema.
- Added restore support for NO-BACKUP tables on Redshift RA3 and Serverless data warehouses. Since these instance types do not support the NO-BACKUP feature, tables marked as NO-BACKUP are always included in snapshots and are now properly restored as permanent tables
- Amazon Redshift now supports writing to external tables when connected using IAM roles or IAM users. Previously, only read operations on external tables were supported with IAM role-based authentication. With this update, Redshift correctly uses your IAM session credentials when writing data to Amazon S3 for external tables, enabling full read and write access.
- For auto refreshing Kafka stream MVs, we now use a more efficient heuristic to determine if there are new records to ingest. This will reduce the amount of data transferred from the Kafka brokers for this check, and reduce the latency on starting the subsequent MV refresh.
- Upgraded backup and restore operations to use FIPS-compliant SHA384 encryption, replacing legacy MD5 hashing
- Enhanced performance for queries combining Lambda User-Defined Functions and UNNEST statements
- Enhanced performance for queries using Lambda User-Defined Functions against system tables and views
- Enhanced query execution performance when using ORDER BY and LIMIT clauses by eliminating unnecessary data scans
- Enhanced Redshift Serverless AI-driven scaling and optimizations with an extended range of supported RPUs, now supporting a minimum of 8 RPUs
- Auto REFRESH queries for Amazon Redshift materialized views (MVs) are now treated as user queries instead of background autonomics processes. Auto REFRESH queries now run with the same priority as other user queries and are no longer deferred by autonomics background processes. The MV Auto REFRESH behavior change feature is only enabled for Amazon Redshift Provisioned clusters on the CURRENT Track of patch release P198 and newer. It is currently disabled on Serverless.
- Improved scheduling for automatic tasks to better prioritize user-critical tasks such as streaming ingestion, auto-copy (COPY JOB), and materialized view auto refresh.
- Fixed an issue where pg_last_query_id() incorrectly returned the rewritten query ID instead of the user query ID for Multi-AZ clusters

- Fixes an issue where SHOW GRANTS would fail to process one-part object references.
- Fixed an issue where DROP TABLE conflicted with the catalog rebuild worker's timestamp auto worker due to simultaneous updates on pg_class_extended, by skipping deletion during DROP TABLE and allowing the timestamp worker to clean up on its next run.
- Fixed zero-ETL channel filter parsing issue when database, schema, or table names contained include/exclude keywords
- Fixed a race condition that could occur when vacuum operations ran concurrently with zero-ETL table operations
- Fixed an issue where zero-ETL integrations incorrectly attempted resync operations instead of transitioning to an error state when KMS encryption context changed
- Fixed a race condition in zero-ETL integrations that could occur during cluster restart while data replication was active
- Fixed an issue where the SHOW TABLES command returned an error when the table's owner had been dropped

Amazon Redshift patch 198

Cluster versions in this patch:

- 1.0.239402 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on March 18, 2026
- 1.0.227967 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on February 25, 2026
- 1.0.224252 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on February 19, 2026
- 1.0.219674 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on February 11, 2026
- 1.0.212832 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on February 10, 2026
- 1.0.204436 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on February 4, 2026

New features and improvements in this patch

- Released a new query compilation enhancement that increases performance for new queries (first runs of queries) in the Redshift CURRENT Maintenance Track. The TRAILING track will follow in an upcoming patch release.
- Improves INTERSECT and EXCEPT queries by applying proper collation rules when comparing columns against constant values.
- Enables auto refresh of manually created materialized views to run on concurrency scaling clusters.
- Improves connection performance by enabling direct connections between main clusters and concurrency scaling clusters, reducing latency and eliminating connection throughput limitations.
- Improves error classification for zero-ETL integrations, making it easier to identify root causes and debug issues.
- Removes error messages from SYS_QUERY_HISTORY for successfully executed queries, providing cleaner query history records.
- Improves compile time logging accuracy for bursted queries in SYS_QUERY_HISTORY, providing better performance insights.
- Adds support for array functions: ARRAY_CONTAINS, ARRAY_POSITION, ARRAYS_OVERLAP, ARRAY_INTERSECTION, ARRAY_SORT, ARRAY_UNION, ARRAY_DISTINCT, ARRAY_EXCEPT, and ARRAY_POSITIONS.
- Adds support for the GET_NUMBER_ATTRIBUTES function that returns the number of attributes within a SUPER object.
- Enables idle transaction timeout for failed transactions to improve resource management.
- Adds support for the TEMPLATE object type that enables you to define reusable formatting parameters, eliminating the need to manually specify them for each operation.
- Improves performance for queries that invoke Lambda User-Defined Functions (LUDFs) on Common Table Expressions (CTEs).
- Improves performance for queries that invoke Lambda User-Defined Functions (LUDFs) in the THEN clause of CASE expressions.
- Creation of new Python User-Defined Functions is deprecated. Existing Python UDFs will be supported until June 30, 2026. Migrate to Lambda User-Defined Functions (LUDFs) for continued support.

- Improves observability for Lambda User-Defined Functions (LUDFs) with a new system table, SYS_LUDF_DETAIL. This table records information and metrics for LUDFs used in queries.
- Auto REFRESH queries for Amazon Redshift materialized views (MVs) are now treated as user queries instead of background autonomics processes. Auto REFRESH queries now run with the same priority as other user queries and are no longer deferred by autonomics background processes. The MV Auto REFRESH behavior change feature is only enabled for Amazon Redshift Provisioned clusters on the CURRENT Track of patch release P198 and newer. It is currently disabled on Serverless.

Amazon Redshift patch 197

Cluster versions in this patch:

- 1.0.237947 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on March 10, 2026
- 1.0.198462 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on January 28, 2026
- 1.0.198462 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on January 21, 2026
- 1.0.194394 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on January 14, 2026
- 1.0.187928 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on January 9, 2026
- 1.0.179517 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on December 16, 2025
- 1.0.176066 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on December 9, 2025
- 1.0.166219 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on November 21, 2025

New features and improvements in this patch

- Global autonomics for clusters in the mesh, Autonomics now consider the workload of the entire mesh to optimize Vacuum, Analyze and ATO.

- Create and refresh Materialized Views from multiple Amazon Redshift data warehouses.
- Added support to store strings literals up to 16,000,000 bytes within the SUPER data types
- Added GET_NUMBER_ATTRIBUTES(super_object) function that returns the total number of attributes contained within a SUPER object.
- Release patch compatibility restrictions are enforced for table restore operations. When attempting to restore a table, the backup can only be used on clusters running the same patch version as when the backup was created, one patch version lower than the backup, or any patch version higher than the backup.
- Enhanced the LIKE operator to properly handle trailing whitespaces in patterns when using CHAR datatype.
- Introduced 4 new spatial H3 functions operating on cell hierarchies: H3_Resolution, H3_ToParent, H3_ToChildren and H3_IsValid.
- Enables Amazon Redshift Federated Permissions that simplifies permissions management across multiple Redshift data warehouses by enabling you to define data permissions once and automatically enforce them across all warehouses in your AWS account.
- Enables object-level grants, fine-grained column-level access control, scoped permissions grants and database-level grants on Amazon Redshift Federated Permissions Catalog objects.
- Enables CREATE, ALTER, ATTACH, DETACH and DROP RLS policy operations on Amazon Redshift Federated Permissions Catalog.
- Enables the ability to turn on or off row-level security for relations in Amazon Redshift Federated Permissions Catalog.
- Enable CREATE, ALTER, ATTACH, DETACH and DROP masking policy operations on Amazon Redshift Federated Permissions Catalog objects.
- Enables SHOW POLICIES to show RLS and Masking policies on a connected database and attachments on relations in Amazon Redshift Federated Permissions Catalog objects.
- Enables database users to assume an IAM role to query on an Amazon Redshift Federated Permissions Catalog objects.
- Enhanced SHOW GRANTS command to support cross-database grant visibility within the same Redshift cluster.
- Added support for SHOW COLUMN GRANTS command to simplify discovery of column-level access controls.
- Added support for SHOW PROCEDURES command to simplify discovery of stored procedure metadata discovery.

- Added support for SHOW FUNCTIONS command to simplify discovery of user-defined function metadata.
- Introduced SHOW PARAMETERS command to display parameter metadata for functions and stored procedures.
- Introduced SHOW CONSTRAINTS command to display constraint metadata for functions and stored procedures.
- Added support for CREATE/CREATE OR REPLACE/DROP VIEW operations on Amazon Redshift Federated Permissions Catalog and on cross-database within the same Redshift cluster.
- Improved constant strings collation handling in some INTERSECT or EXCEPT queries.
- Enhanced SHOW TABLES to display a warning message instead of failing when FAS credentials are missing for AWS Glue Data Catalog access.
- Enhanced SHOW COLUMNS command with additional metadata columns for sort keys, distribution style, encoding, and collation information.
- Added grantor_name column to SHOW GRANTS output and standardized database_name inclusion across all SHOW GRANTS result sets.
- Enhanced SHOW TABLES command with additional columns: table_owner, last_altered_time, last_modified_time, dist_style, and table_sub_type.
- **Identity Center Authentication Support:** Amazon Redshift now supports IAM Identity Center authentication through the new `GetIdentityCenterAuthToken` API operations. This feature enables federated access to Amazon Redshift clusters and serverless workgroups using existing Identity Center identities. Key benefits include single sign-on access using Identity Center credentials, enhanced security with encrypted authentication tokens, comprehensive audit logging for compliance, and seamless integration with existing Identity Center setup. The API operations generate authentication tokens scoped to specific cluster identifiers or workgroup names. Requirements include identity-enhanced credentials with embedded Identity Center identity information and appropriate IAM permissions for API access.

Amazon Redshift patch 196

Cluster versions in this patch:

- 1.0.179872 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on January 8, 2026

- 1.0.163480 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on November 20, 2025
- 1.0.160807 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on November 14, 2025
- 1.0.155905 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on November 11, 2025

New features and improvements in this patch

- Optimized the plans for correlated IN and EXISTS subqueries by decorrelating them into semi joins.
- Extended the SHOW TABLE command to include collation information for every column.

Amazon Redshift patch 195

Cluster versions in this patch:

- 1.0.172541 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on December 8, 2025
- 1.0.162991 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on November 20, 2025
- 1.0.151179 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on November 4, 2025
- 1.0.148180 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on October 31, 2025
- 1.0.142459 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on October 24, 2025

New features and improvements in this patch

- Adds support for writing to Iceberg tables using CREATE TABLE , CREATE TABLE AS SELECT, INSERT INTO, SQL commands.
- Adds support for the SUPER data type in databases with case-insensitive collation.
- Updates the IANA Time Zone Database database version to 2025b.

- Adds support for the TZDB_VERSION function. This function displays the current IANA Time Zone Database version in use.
- Amazon Redshift now enforces patch compatibility restrictions. Backups created on patch versions that are more than one patch version ahead of the current cluster are no longer permitted to perform table restore operations.
- Extends UNNEST functionality to support JOIN expressions. Previously, UNNEST was limited to single-table references.
- Adds support for the JIT (Just In Time) ANALYZE feature. JIT ANALYZE automatically, intelligently, and efficiently collects the minimal set of statistics that Amazon Redshift's query optimizer needs to generate efficient query execution plans, allowing customers to achieve best price-performance for Apache Iceberg out-of-box (OOB) queries and workloads.

Amazon Redshift patch 194

Cluster versions in this patch:

- 1.0.148169 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on November 5, 2025
- 1.0.138443 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on October 18, 2025
- 1.0.129791 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on October 8, 2025

New features and improvements in this patch

- Amazon Redshift will no longer fail a query that accesses an AWS Lake Formation-governed relation where the `AuthorizedColumns` field includes a backslash.
- Added support for the NULL AS option in COPY command for SUPER columns, letting you specify a custom string to represent NULL values.
- Added support for the ISOWEEK and ISOYEAR date parts in EXTRACT and DATE_PART functions when used with dates and timestamps.
- Fixed PartiQL navigation on grouped variables to ensure correct parsing and evaluation of nested field access in GROUP BY queries.

- Added support to view primary key and foreign key constraints using the SHOW CONSTRAINTS command.
- Added support for local materialized views created from data sharing materialized views.
- Improves the performance of queries that retrieve data from different tables and invoke Lambda User Defined Functions in join conditions.
- Add support for MVs on Datasharing MVs
- The DROP DATABASE command now uses a background worker to remove leader node catalog directories after commit. Using a worker avoids rare bugs related to catalog directory cleanup.

Amazon Redshift patch 193

Cluster versions in this patch:

- 1.0.136890 – **TRAILING Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on October 17, 2025
- 1.0.127211 – **CURRENT Track** Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on September 30, 2025

New features and improvements in this patch

- Added support for creating tables using CREATE TABLE LIKE to clone one parent table from remote databases (both within the same cluster and between different clusters).
- Added ST_GeomFromGeoJSON overload that creates a geometry from the SUPER datatype in GeoJSON format, with support for geometries up to 1 MB.
- Changed the default fallback option in case of misconfigurations in automatic workload management (WLM). The default fallback option is now automatic WLM instead of manual WLM.
- Updated time zone calculation by adopting the latest IANA Time Zone Database patches. This change alters how date and time conversions function for certain time zones and time periods. For more information, see Amazon Redshift uses up-to-date IANA Time Zone Database.
- Added support for multidimensional data layout (MDDL) sorting. MDDL sort keys dynamically sort data based on actual query filters, accelerating query performance. For more information, see Multidimensional data layout storing in the *Amazon Redshift Database Developer Guide*.

Amazon Redshift patch 192

Cluster versions in this patch:

- 1.0.124422 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on September 22, 2025
- 1.0.122914 – Trailing Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on September 16, 2025
- 1.0.121035 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on August 22, 2025
- 1.0.119650 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on August 7, 2025
- 1.0.118447 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on July 24, 2025

New features and improvements in this patch

- Fixes an issue when JOIN on a collated case-insensitive column and a constant string column selects a mismatching hash function.

Amazon Redshift patch 191

Cluster versions in this patch:

- 1.0.117891 – Trailing Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on July 29, 2025
- 1.0.117891 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on July 22, 2025
- 1.0.117458 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on July 15, 2025
- 1.0.116415 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on June 26, 2025
- 1.0.116263 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on June 25, 2025

- 1.0.115865 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on June 18, 2025

New features and improvements in this patch

- Adds support for the UNNEST operation in the SELECT statement's FROM clause, enabling customers to convert array elements into individual rows.
- Improves error handling for PartiQL unnesting syntax when using SUPER data type within Common Table Expressions (CTEs) and recursive CTEs, particularly when resolving table aliases with identical names.
- Improves performance for automatic analyze by adding support for incremental automatic analyze for large tables.
- Amazon Redshift now supports Cascading Refresh of nested Materialized Views (MVs).
- Amazon Redshift now supports Auto Refresh of Materialized Views (MVs) defined on Apache Iceberg tables.
- Adds support for Amazon RDS for Oracle zero-ETL integrations with Amazon Redshift.
- Adds support for Amazon RDS for PostgreSQL zero-ETL integrations with Amazon Redshift.

Amazon Redshift patch 190

Cluster versions in this patch:

- 1.0.115140 – Trailing Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on June 16, 2025
- 1.0.115564 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on June 14, 2025
- 1.0.113436 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on May 15, 2025
- 1.0.112895 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on May 7, 2025

New features and improvements in this patch

- Adds improved models for Amazon Redshift Serverless AI-driven scaling and optimizations.

- Adds support for auto-copy when using Amazon Redshift with enhanced VPC routing enabled.
- Adds support for multiple automatic table optimization operations to run concurrently across different tables.
- Fixes a rare condition in streaming ingestion where queued autorefresh queries trigger UNDO queries that also queue, leading to lock contention.
- Adds support for Oracle Database@AWS zero-ETL integrations with Amazon Redshift.

Amazon Redshift patch 189

Cluster versions in this patch:

- 1.0.112790 – Trailing Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on May 14, 2025
- 1.0.112790 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on May 6, 2025
- 1.0.112086 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on April 29, 2025
- 1.0.111040 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on April 16, 2025
- 1.0.109768 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on March 22, 2025
- 1.0.109284 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on March 13, 2025
- 1.0.108971 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on March 7, 2025

New features and improvements in this patch

- Adds support for concurrency scaling for write queries using the SUPER, GEOMETRY, and GEOGRAPHY data types.
- Fixes an issue that allowed SUPER, GEOMETRY, or GEOGRAPHY columns to be set as DISTKEY or SORTKEY columns during table creation with CREATE TABLE AS.
- Adds support for the TRY_CAST function.
- Enables support for running multiple vacuum commands concurrently across different tables.

Amazon Redshift patch 188

Cluster versions in this patch:

- 1.0.109616 – Trailing Amazon Redshift Serverless workgroup version – Released on March 27, 2025
- 1.0.109616 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on March 24, 2025
- 1.0.108470 – Trailing Amazon Redshift Serverless workgroup version – Released on March 13, 2025
- 1.0.108470 – Current Amazon Redshift Serverless workgroup version – Released on March 11, 2025
- 1.0.108950 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on March 10, 2025
- 1.0.108790 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on March 9, 2025
- 1.0.108470 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on March 4, 2025
- 1.0.107910 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on February 20, 2025
- 1.0.107360 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on February 13, 2025
- 1.0.106767 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on February 5, 2025

New features and improvements in this patch

- Adds support for automatic mounting of an Amazon S3 Tables catalog, making it easier for you to run queries on Apache Iceberg tables managed in Amazon S3 Tables. See Prerequisites for managing Amazon Redshift namespaces in the AWS Glue Data Catalog in the *AWS Lake Formation Developer Guide* for required permissions.
- You can now bypass the data masking check and add a table with a dynamic data masking (DDM) attachment to a datashare.

- Disallows the use of correlation references that skip a query block, also known as "skip-level correlation references". For more information, see Correlated subquery patterns that are not supported in the *Amazon Redshift Database Developer Guide*.
- Adds support for correlated SUPER data type queries with data sharing and 3-part notation.
- Adds support for the EXCLUDE SQL keyword.
- Adds support for new GROUP BY ALL SQL keyword.
- Improves performance for workload management (WLM) queues configured with roles, resulting in lower lock contention.
- Adds support for automatically refreshed materialize views on zero-ETL tables with history mode turned on.
- Adds support for null characters embedded in strings during zero-ETL replication.
- Improves models for AI driven scaling and optimizations.
- Adds the `username` column to the `sys_query_history` view.

Amazon Redshift patch 187

Cluster versions in this patch:

- 1.0.107351 – Amazon Redshift provisioned cluster trailing track version and Amazon Redshift Serverless workgroup version – Released on February 13, 2025
- 1.0.106452 – Amazon Redshift provisioned cluster trailing track version – Released on February 5, 2025
- 1.0.106980 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on February 3, 2025
- 1.0.106452 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on January 24, 2025
- 1.0.106073 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on January 21, 2025
- 1.0.105722 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on January 10, 2025
- 1.0.105373 – Current Amazon Redshift provisioned cluster version and Amazon Redshift Serverless workgroup version – Released on January 8, 2025
- 1.0.104930 – Amazon Redshift Serverless workgroup version – Released on December 20, 2024

New features and improvements in this patch

- Introduces two new spatial H3 functions (H3_center and H3_boundary).
- Enables dynamic disk cache on concurrency scaling clusters.
- Fixes an inefficiency in memory usage during the ingestion of sorted tables that could impact performance.
- Fixes an issue where certain queries with a subquery inside the SELECT clause, such as `SELECT ARRAY_FLATTEN((SELECT FROM ...))`, would incorrectly trigger a `size >= min_partiql_size` XCHECK error in specific scenarios involving a single SUPER value.
- Fixes an issue where non-nullable SUPER expressions (`json_serialize()`, `json_size()`, `json_typeof()`, `is_object()`, and several others) would sometimes produce erroneous results when combined with argument expressions such as `CASE ... END` or `COALESCE()`.
- Fixes an issue where SELECT from late binding views would raise an ERROR after enabling row-level security on the view.
- Fixes an issue where CDC on a zero-ETL integration lead to a high leader node CPU consumption.
- Adds the capability to zero-ETL integrations to query tables in all states, including during updates.
- Adds the capability to zero-ETL integrations to truncate large texts/strings for mapping to the max varchar size on Amazon Redshift.
- Adds the capability to zero-ETL integrations to replace invalid UTF-8 characters with a specified character of your choice.
- Fixes an issue on 'refresh interval' for zero-ETL integration with Amazon RDS for MySQL.
- Adds support for JSON/JSONB data type for zero-ETL integration with Aurora PostgreSQL.
- Fixes an issue where queries containing expressions with `IS [NOT] {TRUE|FALSE|UNKNOWN}` would encounter an assertion error.
- Fixes an issue where the `json_extract_path_text()` and `json_extract_array_element_text()` functions would produce an empty string `''` instead of `NULL` or vice versa in cases like accessing a non-existent array element or attribute, the JSON `'null'` value, or an empty JSON string.
- Improves performance of INSERT/COPY statements in concurrent transactions writing to the same table by allowing them to share a lock and make progress until they need to write data to the table.

Amazon Redshift patch 186

Cluster versions in this patch:

- 1.0.82096 – Trailing track version – Released on January 10, 2025
- 1.0.82000 – Amazon Redshift Serverless version – Released on January 10, 2025
- 1.0.81981 – Current track version – Released on January 10, 2025
- 1.0.81475 – Trailing track version – Released on January 6, 2025
- 1.0.81473 – Amazon Redshift Serverless version – Released on January 6, 2025
- 1.0.81462 – Current track version – Released on January 6, 2025
- 1.0.80643 – Trailing track version – Released on December 17, 2024
- 1.0.80583 – Amazon Redshift Serverless version – Released on December 17, 2024
- 1.0.80560 – Current track version – Released on December 17, 2024
- 1.0.80498 – Amazon Redshift Serverless version – Released on December 13, 2024
- 1.0.80491 – Current track version – Released on December 13, 2024
- 1.0.80036 – Amazon Redshift Serverless version – Released on December 6, 2024
- 1.0.80009 – Current track version – Released on December 6, 2024
- 1.0.79372 – Amazon Redshift Serverless version – Released on November 26, 2024
- 1.0.79237 – Amazon Redshift Serverless version – Released on November 24, 2024
- 1.0.79229 – Current track version – Released on November 24, 2024
- 1.0.79003 – Amazon Redshift Serverless version – Released on November 19, 2024
- 1.0.78987 – Current track version – Released on November 19, 2024
- 1.0.78890 – Amazon Redshift Serverless version – Released on November 18, 2024
- 1.0.78881 – Current track version – Released on November 18, 2024
- 1.0.78646 – Amazon Redshift Serverless version – Released on November 14, 2024
- 1.0.78641 – Current track version – Released on November 14, 2024
- 1.0.78178 – Amazon Redshift Serverless version – Released on November 12, 2024
- 1.0.78160 – Current track version – Released on November 12, 2024
- 1.0.77809 – Amazon Redshift Serverless version – Released on October 31, 2024

- 1.0.77777 – Current track version – Released on October 31, 2024
- 1.0.77292 – Amazon Redshift Serverless version – Released on October 24, 2024
- 1.0.77272 – Current track version – Released on October 24, 2024
- 1.0.77040 – Amazon Redshift Serverless version – Released on October 22, 2024
- 1.0.77028 – Current track version – Released on October 22, 2024

New features and improvements in this patch

- Adds support for automatic and incremental refresh of materialized views on tables from zero-ETL integrations with Amazon Aurora MySQL, Amazon Aurora PostgreSQL, Amazon RDS for MySQL, and Amazon DynamoDB.
- Improves how Amazon Redshift rewrites queries with correlated single-row subqueries, such as `SELECT … WHERE key = (SELECT … correlated subquery)`. Note that such queries are only valid if the subquery yields a single row. Due to the improved rewrite, certain queries that violate this condition might now fail with ERROR: single-row subquery returns more than one row in cases where they were previously permitted. To avoid this, such subqueries might need to be corrected so that they are guaranteed to return a single row, for example, by adding a MIN() or MAX() aggregate.
- Adds a SQL Identifier 'KAFKA' in Amazon Redshift to support streaming from external Kafka sources into Amazon Redshift with direct streaming ingestion. These sources include external Kafka sources such as Confluent Managed Cloud and Apache Kafka.
- Adds support for multi-warehouse writes using data sharing so that you can scale your write workloads and achieve better performance for extract, transform, and load (ETL) workloads by using different warehouses of various types and sizes, based on your workload needs.
- You can now run `SELECT` queries on Apache Iceberg tables managed in Amazon S3 Tables using Amazon Redshift For more information, see Accessing Amazon S3 tables with Amazon Redshift in the *Amazon Simple Storage Service User Guide*.
- Adds support for cross-database write queries in Amazon Redshift, so that you can write across multiple databases in an Amazon Redshift cluster. For more information, see Cross-database queries.

Amazon Redshift patch 185

Cluster versions in this patch:

- 1.0.79878 – Trailing track version – Released on December 3, 2024
- 1.0.79868 – Amazon Redshift Serverless version – Released on December 3, 2024
- 1.0.79845 – Current track version – Released on December 3, 2024
- 1.0.78946 – Trailing track version – Released on November 19, 2024
- 1.0.78354 – Trailing track version – Released on November 12, 2024
- 1.0.78130 – Amazon Redshift Serverless version – Released on November 12, 2024
- 1.0.78125 – Current track version – Released on November 12, 2024
- 1.0.78016 – Amazon Redshift Serverless version – Released on November 5, 2024
- 1.0.78014 – Current track version – Released on November 5, 2024
- 1.0.77707 – Amazon Redshift Serverless version – Released on November 4, 2024
- 1.0.77687 – Current track version – Released on November 4, 2024
- 1.0.77467 – Amazon Redshift Serverless version – Released on November 4, 2024
- 1.0.77433 – Current track version – Released on November 4, 2024
- 1.0.77467 – Amazon Redshift Serverless version – Released on October 29, 2024
- 1.0.77433 – Current track version – Released on October 29, 2024
- 1.0.76991 – Amazon Redshift Serverless version – Released on October 21, 2024
- 1.0.76913 – Current track version – Released on October 21, 2024
- 1.0.76645 – Amazon Redshift Serverless version – Released on October 14, 2024
- 1.0.76642 – Current track version – Released on October 14, 2024
- 1.0.76242 – Amazon Redshift Serverless version – Released on October 10, 2024
- 1.0.76230 – Current track version – Released on October 10, 2024

New features and improvements in this patch

- Adds support for incremental refresh of materialized views (MVs) created on data lake tables.
- Introduces support to alter the distribution key and sort key for materialized views.
- Adds support for the integration of Amazon Redshift machine learning (ML) with Amazon Bedrock to leverage large language models (LLMs) from simple SQL commands along with the data in Amazon Redshift.
- Fixes a bug that allowed the processing of empty shapefiles in Amazon Redshift spatial data without throwing errors.

- Fixes a bug that allowed varbit and varbinary datatypes to ingest an empty string value as "", rather than NULL, in a zero-ETL integration.
- Fixes a race condition with a result cache, which caused cross-database queries to return stale results in a zero-ETL integration.
- Optimizes the zero-ETL integration resync process, resulting in shorter resync times.
- Improves zero-ETL bootstrap time after recovery and system mainteanance operations. Now the system can recover the CDC after the last query operation, instead of being able to recover up until the latest available CDC point.
- Improves observability for zero-ETL integration with a new system table, `sys_integration_table_activity`. This table tracks zero-ETL integration table inserts, updates, and deletes.
- Improves the auto-creation of federated roles experience. Amazon Redshift administrators now have more control over the auto-creation of federated roles during federated user login. You can now to enable, disable, apply filters to, and configure auto-creation settings for each identity provider.
- IAM Identity Center users can run COPY, UNLOAD, and CREATE LIBRARY with IAM Identity Center credentials using S3 access grants.
- Auto-copy (COPY JOB) that enables continuous ingestion of files from Amazon S3, is now generally available.

Amazon Redshift patch 184

Cluster versions in this patch:

- 1.0.77706 – Trailing track version – Released on November 12, 2024
- 1.0.76832 – Trailing track version – Released on October 17, 2024
- 1.0.76169 – Amazon Redshift Serverless version – Released on October 10, 2024
- 1.0.76142 – Current track version – Released on October 10, 2024
- 1.0.75677 – Amazon Redshift Serverless version – Released on September 27, 2024
- 1.0.75672 – Current track version – Released on September 27, 2024
- 1.0.75504 – Amazon Redshift Serverless version – Released on September 23, 2024
- 1.0.75449 – Current track version – Released on September 24, 2024
- 1.0.74765 – Amazon Redshift Serverless version – Released on September 12, 2024

- 1.0.74754 – Current track version – Released on September 12, 2024

New features and improvements in this patch

- Amazon Redshift streaming materialized views (MVs) for streaming data ingestion have increased VARBYTE column sizes up from 1,024,000 bytes. Amazon Redshift can now ingest records from Amazon Kinesis Data Streams up to 1,048,576 bytes, or 1MiB. Redshift can ingest records from Amazon Managed Streaming for Apache Kafka up to 16,777,216 bytes, or 16 MiB. If you're using the ATA SQL command `ALTER TABLE <target_tbl> APPEND FROM <streaming_mv>`, please drop and recreate your <target_tbl> to have the corresponding larger VARBYTE column sizes.
- Amazon Redshift datashares now can include Amazon S3 data lake tables and views that reference AWS Glue Data Catalog including tables governed by Lake Formation.
- Adds support for automatic and incremental refresh of materialized views on tables from zero-ETL integration with DynamoDB.
- Adds support to configure the 'refresh interval' on zero-ETL integration tables to specify the replication refresh rate on Amazon Redshift. You can set it at the time of creating a database for a new integration or alter the database of an existing integration.
- Adds support for Mutual Transport Layer Security (mTLS) authentication in Amazon Redshift streaming ingestion for Amazon Managed Streaming for Apache Kafka.
- Adds support for query hash, a unique identifier for a SQL query based on the textual representation of the query and the values of its parameters. It can be used to identify, group, and analyze similar queries. Query hash can now be found in SYS_QUERY_HISTORY view, with the addition of two new columns:
  - `user_query_hash` – The hash as submitted by the user including the query literals.
  - `generic_query_hash` - The hash as submitted by the user without any query literals.
- Fixes system deadlock in a rare case where zero-ETL was running CDC replication and scan, performing table queries with result cache.
- Addresses an issue in Workload Management (WLM) where a Python User Defined Function (UDF) queries would preempt other queries when WLM resources for Python UDF queries were unavailable.
- Addresses an issue where Workload Management (WLM) would fail to route queries to the queue mapped to a newly created user role.
- Improves disk utilization on smaller data sharing consumers querying large producer tables.

- Improves performance of INSERT statements for provisioned clusters elastically resized to a higher size.

Amazon Redshift patch 183

Cluster versions in this patch:

- 1.0.75655 – Trailing track version – Released on September 30, 2024
- 1.0.75388 – Amazon Redshift Serverless version – Released on September 25, 2024
- 1.0.75379 – Current track version – Released on September 25, 2024
- 1.0.74967 – Amazon Redshift Serverless version – Released on September 17, 2024
- 1.0.74927 – Current track version – Released on September 17, 2024
- 1.0.74518 – Amazon Redshift Serverless version – Released on September 11, 2024
- 1.0.74503 – Current track version – Released on September 11, 2024
- 1.0.74223 – Amazon Redshift Serverless version – Released on September 5, 2024
- 1.0.74159 – Current track version – Released on September 5, 2024
- 1.0.74126 – Amazon Redshift Serverless version – Released on August 30, 2024
- 1.0.74097 – Current track version – Released on August 30, 2024
- 1.0.73016 – Amazon Redshift Serverless version – Released on August 8, 2024
- 1.0.72982 – Current track version – Released on August 8, 2024

New features and improvements in this patch

- Adds support for discovery of Scoped Permissions via SVV_DATABASE_PRIVILEGES and SVV_SCHEMA_PRIVILEGES. Also introduces the column privilege_scope to SVV_DATABASE_PRIVILEGES and SVV_SCHEMA_PRIVILEGES.
- Improves performance of queries that execute distinct aggregation operations when grouping columns have a low number of distinct values (NDV).
- Improves performance of INSERT/COPY statements for provisioned data warehouses elastically resized by 2x or higher in size.
- Supports session context variables inside a Dynamic Data Masking policy.
- Adds support for subqueries and views as the source for the MERGE statement.

- Supports stored procedures containing a MERGE statement on provisioned concurrency scaling and serverless autoscaling compute.
- Improves query performance with better resource prediction in workload management for COPY commands and for warehouses which undergo resizes.
- Improves resilience to out of memory errors in clusters with limited memory available
- Adds support for non-ASCII characters as field delimiters to the COPY command.
- Adds support for ingesting data encoded in the ISO-8859-1 character set using the COPY command.
- Removes requirement to specify CLUSTER_ARN in MSK external schema definition if specifying URI.
- Supports applying range scan filters during scans on zero-ETL integration tables.
- Supports adding sort keys to zero-ETL integration tables.
- Supports database options like `serializable` and `collation` to be specified with CREATE DATABASE statement when creating a zero-ETL integration database.
- Fixed the issue that was causing the cluster to restart when the data filter exceeded 2 KB in a zero-ETL integration.

Amazon Redshift patch 182

Cluster versions in this patch:

- 1.0.73589 – Trailing track version – Released on August 22, 2024
- 1.0.73359 – Amazon Redshift Serverless version – Released on August 15, 2024
- 1.0.73348 – Current track version – Released on August 15, 2024
- 1.0.72917 – Amazon Redshift Serverless version – Released on August 12, 2024
- 1.0.72899 – Current track version – Released on August 12, 2024
- 1.0.72528 – Amazon Redshift Serverless version – Released on August 7, 2024
- 1.0.72503 – Current track version – Released on August 8, 2024
- 1.0.72239 – Amazon Redshift Serverless version – Released on August 1, 2024
- 1.0.71714 – Amazon Redshift Serverless version – Released on July 24, 2024
- 1.0.71629 – Current track version – Released on July 24, 2024
- 1.0.70953 – Amazon Redshift Serverless version – Released on July 11, 2024

- 1.0.70890 – Current track version – Released on July 11, 2024
- 1.0.70716 – Amazon Redshift Serverless version – Released on July 8, 2024
- 1.0.70695 – Current track version – Released on July 8, 2024
- 1.0.69945 – Amazon Redshift Serverless version – Released on June 27, 2024
- 1.0.69938 – Current track version – Released on June 27, 2024

New features and improvements in this patch

- Updates LISTAGG, MEDIAN, PERCENTILE_CONT and PERCENTILE_DISC to no longer require user-defined tables. Queries that reference catalog tables or that don't reference any tables can also use these functions.
- Reduces query-planning time for datasharing read queries by consolidating temp tables across multiple queries within a single session, for high-concurrency workloads.
- Provides general availability of Redshift machine learning (ML) integration with Amazon SageMaker AI Jumpstart, for bringing your own large-language models.
- Introduces support for the SUPER input and output data type in Redshift ML.
- Enables support for an UPDATE statement with a JOIN clause when the target table is protected by a dynamic data-masking policy and referenced in the JOIN clause.
- Allows querying of a zero-ETL database on Redshift, even after the integration is deleted from source.
- Fixes a replication error that could cause zero-ETL integration to fail. This makes an integration more resilient.
- Allows users other than the user who created a zero-ETL integration to query the data, after GRANT permissions.
- Fixes an out-of-memory issue that could cause cluster restarts in an Amazon Redshift provisioned cluster with zero-ETL integration enabled.
- Enables creation of an RDS for MySQL zero-ETL integration with Redshift, from a source RDS Multi-AZ DB cluster. A Multi-AZ DB cluster is a semisynchronous, high-availability deployment mode of Amazon RDS with two readable replica DB instances.
- Enables a user to connect to an Amazon MSK cluster from an Amazon Redshift streaming consumer client by specifying the Amazon MSK cluster's broker URI in the external schema definition needed to associate an Amazon Redshift streaming materialized view with an Amazon MSK topic. This feature removes the need to obtain the Amazon MSK bootstrap broker node

name by calling the GetBootStrapBroker API on the Amazon MSK cluster over an internet gateway.

- Resolves the issue with resuming Amazon Redshift Serverless instances, enabling an existing database user to resume a Serverless instance when connecting to databases in Amazon Redshift query editor v2, using the IAM Identity Center authentication method.
- Optimizes CDC replication and reduced resource utilization on Redshift compute by moving to table-based sharding.
- Improves query performance using enhanced resource prediction in workload management (WLM).
- Fixes queries failing on concurrency scaling clusters with the message: ran out of WLM queues for restart.
- Fixes a workload management (WLM) issue where Amazon Redshift falls back to manual WLM when customers try to apply an invalid WLM configuration.
- Allows data sharing consumers to run read queries even when the producer is down due to planned maintenance or an unplanned outage.
- Fixes a rare cluster restart issue that occurs when the ANY_VALUE function is used in queries that aggregate data, for example, the COUNT(DISTINCT) aggregation function.

Amazon Redshift patch 181

Cluster versions in this patch:

- 1.0.72031 – Current track version – Released on August 1, 2024
- 1.0.71912 – Trailing track version – Released on August 1, 2024
- 1.0.70665 – Amazon Redshift Serverless version – Released on July 8, 2024
- 1.0.70634 – Current track version – Released on July 8, 2024
- 1.0.69954 – Amazon Redshift Serverless version – Released on June 26, 2024
- 1.0.69952 – Current track version – Released on June 26, 2024
- 1.0.69497 – Amazon Redshift Serverless version – Released on June 18, 2024
- 1.0.69451 – Current track version – Released on June 18, 2024
- 1.0.69076 – Amazon Redshift Serverless version – Released on June 14, 2024
- 1.0.69065 – Current track version – Released on June 14, 2024

- 1.0.68555 – Amazon Redshift Serverless version – Released on May 31, 2024
- 1.0.68540 – Current track version – Released on May 31, 2024
- 1.0.68328 – Amazon Redshift Serverless version – Released on May 23, 2024
- 1.0.68205 – Current track version – Released on May 23, 2024
- 1.0.67796 – Amazon Redshift Serverless version – Released on May 15, 2024
- 1.0.67788 – Current track version – Released on May 15, 2024
- 1.0.67308 – Amazon Redshift Serverless version – Released on May 1, 2024
- 1.0.67305 – Current track version – Released on May 1, 2024

New features and improvements in this patch

- Introduces support for the 'lower_attribute_names()' and 'upper_attribute_names()' functions which modify the case of attribute names for SUPER object values.
- Fixes an issue in CREATE TABLE LIKE when using an identity column. Previously, the new table would inherit the identifier from the source table. This caused problems if the source table was later dropped, since the identifier would become invalid in the new table.
- Fixes an issue preventing some external tables from showing in SVV_ALL_TABLES.
- Improves cluster bootstrap time, and speeds up query initialization for high concurrent workloads.
- Fixes an issue with federated query that caused errors when passing split_part() functions to the federated source to RDS and Aurora MySQL
- Supports user initiated changes to the distribution key through ALTER TABLE...ALTER DISTSTYLE KEY DISTKEY commands on provisioned concurrency scaling clusters and serverless autoscaling compute.
- Supports manually refreshed materialized views that involve aggregation on provisioned concurrency scaling and serverless autoscaling compute.
- Adds support for zero-ETL to handle records up to 16 MB in size and for supporting SUPER values up to 16 MB.
- Enhances error messages during initial sync in zero-ETL from Aurora MySQL by providing additional details like schema and table name.
- Introduces support for tagging with Amazon Redshift ML CREATE MODEL. With this improvement, you can now tag Amazon SageMaker AI resources used by Amazon Redshift ML. Tagging helps you manage, identify, organize, search for, and filter resources.

- Improves the performance of queries involving Lambda user-defined functions (UDFs) by optimizing the data processing with the AWS Lambda.
- Reduces memory utilization during data ingestion in sorted tables of elastically resized and serverless clusters.
- Adds support for newlines (\n) in column `query_text` in view SYS_QUERY_HISTORY and for column `text` in view SYS_QUERY_TEXT.

Amazon Redshift patch 180

Cluster versions in this patch:

- 1.0.68520 – Trailing track version – Released on May 28, 2024
- 1.0.67699 – Trailing track version – Released on May 15, 2024
- 1.0.66960 – Trailing track version – Released on April 21, 2024
- 1.0.66954 – Current track version – Released on April 21, 2024
- 1.0.66276 – Current track version – Released on April 12, 2024
- 1.0.66290 – Amazon Redshift Serverless version – Released on April 10, 2024
- 1.0.63590 – Current track version – Released on February 19, 2024
- 1.0.63567 – Amazon Redshift Serverless version – Released on February 16, 2024
- 1.0.63282 – Amazon Redshift Serverless version – Released on February 13, 2024
- 1.0.63269 – Current track version – Released on February 13, 2024
- 1.0.63215 – Amazon Redshift Serverless version – Released on February 12, 2024
- 1.0.63205 – Current track version – Released on February 12, 2024
- 1.0.63030 – Amazon Redshift Serverless version – Released on February 7, 2024
- 1.0.62913 – Current track version – Released on February 7, 2024
- 1.0.62922 – Amazon Redshift Serverless version – Released on February 5, 2024
- 1.0.62878 – Current track version – Released on February 5, 2024
- 1.0.62698 – Amazon Redshift Serverless version – Released on January 31, 2024
- 1.0.62614 – Current track version – Released on January 31, 2024
- 1.0.61687 – Amazon Redshift Serverless version – Released on January 5, 2024
- 1.0.61678 – Current track version – Released on January 5, 2024

- 1.0.61567 – Amazon Redshift Serverless version – Released on December 31, 2023
- 1.0.61559 – Current track version – Released on December 31, 2023
- 1.0.61430 – Amazon Redshift Serverless version – Released on December 29, 2023
- 1.0.61395 – Current track version – Released on December 29, 2023

New features and improvements in this patch

- Changes CURRENT_USER to no longer truncate the returned username to 64 characters.
- Adds the ability to apply data masking policies on standard views and late binding views.
- Adds the ability to apply dynamic data masking (DDM) to scalar attributes in SUPER data type columns.
- Adds OBJECT_TRANSFORM SQL function. For more information, see OBJECT_TRANSFORM function in the *Amazon Redshift Database Developer Guide*.
- Adds the ability to apply AWS Lake Formation fine-grained access control to your nested data, and query with Amazon Redshift data lake analytics.
- Adds the INTERVAL data type.
- Adds CONTINUE_HANDLER, which is a type of exception handler that controls the flow of a stored procedure. Using it, you can catch and handle exceptions without ending the existing statement block.
- Adds the ability to define permissions on a scope (schema or database) in addition to individual objects. This allows users and roles to be granted a permission on all current and future objects within the scope.
- Adds the ability to create a database from a datashare with permissions that let consumer-side administrators grant individual permissions on shared database objects to consumer-side users and roles.
- Adds support for the SUPER return data type from remote BYOM models. This expands the range of accepted SageMaker AI models to include those with more complex return formats.
- Changes external functions to now implicitly cast numbers with or without fractional parts to the numeric data type of the column. For int2, int4, and int8 columns, numbers with fractional digits are accepted by truncating unless the number is out of range. For float4 and float8 columns, numbers are accepted without fractional digits.
- Adds three spatial functions that work with the H3 hierarchical geospatial indexing grid system: H3_FromLongLat, H3_FromPoint, and H3_Polyfill.

Amazon Redshift patch 179

Cluster versions in this patch:

- 1.0.62317 – Amazon Redshift Serverless version – Released on January 29, 2024
- 1.0.62312 – Trailing track version – Released on January 29, 2024
- 1.0.61631 – Amazon Redshift Serverless version – Released on January 5, 2024
- 1.0.61626 – Current track version – Released on January 5, 2024
- 1.0.61191 – Current track version – Released on December 16, 2023
- 1.0.61150 – Amazon Redshift Serverless version – Released on December 16, 2023
- 1.0.60982 – Amazon Redshift Serverless version – Released on December 13, 2023
- 1.0.60854 – Current track version – Released on December 10, 2023
- 1.0.60354 – Amazon Redshift Serverless version – Released on November 22nd, 2023
- 1.0.60353 – Current track version – Released on November 21st, 2023
- 1.0.60293 – Amazon Redshift Serverless version – Released on November 21st, 2023
- 1.0.60292 – Current track version – Released on November 22nd, 2023
- 1.0.60161 – Amazon Redshift Serverless version – Released on November 18th, 2023
- 1.0.60140 – Current track version – Released on November 18th, 2023
- 1.0.60139 – Amazon Redshift Serverless version – Released on November 18th, 2023
- 1.0.59947 – Amazon Redshift Serverless version – Released on November 16th, 2023
- 1.0.59945 – Current track version – Released on November 16th, 2023
- 1.0.59118 – Amazon Redshift Serverless version – Released on November 9th, 2023
- 1.0.59117 – Current track version – Released on November 9th, 2023

New features and improvements in this patch

- Adds support so that federated users with appropriate permissions can view row-level security and dynamic data masking system views, including:
  - SVV_ATTACHED_MASKING_POLICY
  - SVV_MASKING_POLICY
  - SVV_RLS_ATTACHED_POLICY
  - SVV_RLS_POLICY

  - SVV_RLS_RELATION
- Adds functionality such that a query that contains only scalar functions in the FROM clause now results in an error.
- Adds CREATE TABLE AS (CTAS) statements with permanent target tables functionality to concurrency scaling clusters. Concurrency scaling clusters now support more queries.
- Adds the following system tables to track table redistribution status after running classic resize on RA3 clusters:
  - The SYS_RESTORE_STATE system table shows table level redistribution progress.
  - The SYS_RESTORE_LOG system table shows historical throughput of data redistribution.
- Improves slice skew minimizing on EVEN tables after running classic resize on RA3 node types. This is also applicable to patch 178 clusters that ran classic resize.
- Adds support for UNLOAD with EXTENSION on concurrency scaling clusters.
- Improves performance for queries that contain Λ UDFs in HashJoins and NestLoop joins.
- Improves performance of Elastic Resize on RA3 node types.
- Improves performance for data sharing queries.
- Improves performance of manually initiated analyze queries in elastic-resized provisioned clusters and serverless workgroup.
- Improves auto WLM query performance with better resource prediction in workload management.
- Removes the functionality of launching clusters into dedicated tenancy VPCs. This change doesn't affect tenancy of any EC2 instances in the VPC. You can modify tenancy of your VPC to default with the `modify-vpc-tenancy` AWS CLI command.
- Materialized view manual refresh is now supported on provisioned concurrency scaling clusters and serverless autoscaling compute.
- Adds support for INTERVAL literals to the EXTRACT function. For example, `EXTRACT('hours' from Interval '50 hours')` returns 2 because 50 hours is interpreted as 2 days and 2 hours, and the hour component of 2 is extracted.

Amazon Redshift patch 178

Cluster versions in this patch:

- 1.0.63327 - Current track version – Released on February 9, 2024

- 1.0.63313 - Trailing track version – Released on February 9, 2024
- 1.0.60977 - Trailing track version – Released on December 15, 2023
- 1.0.59596 - Current track version – Released on November 9th, 2023
- 1.0.58593 - Amazon Redshift Serverless version – Released on October 23rd, 2023
- 1.0.58558 - Current track version – Released on October 23rd, 2023
- 1.0.57864 - Current track version – Released on October 12th, 2023
- 1.0.57850 - Amazon Redshift Serverless version – Released on October 12th, 2023
- 1.0.56952 - Current track version – Released on September 25th, 2023
- 1.0.56970 - Amazon Redshift Serverless version – Released on September 25th, 2023

New features and improvements in this patch

- Amazon Redshift now has improved data sharing query performance by speeding up metadata refresh on consumer instances while concurrent data changes are happening on the producer instance.
- Adds support for automatic and incremental refresh of materialized views on Amazon Redshift data sharing consumer instances when the base tables of the materialized view refer to the shared data.
- Adds support for storing large objects up to 16 MB in size in the SUPER data type. When ingesting from JSON, PARQUET, TEXT, and CSV source files, you can load semi-structured data or documents as values in the SUPER data type, up to 16 MB.
- Adds support for elastic resize for scaling to and from a single-node Amazon Redshift RA3 cluster.
- Single-node Amazon Redshift RA3 clusters now can now benefit from encryption enhancements, reducing the overall encryption time and improving the availability of the data warehouse during the encryption process.
- Improves support for queries when unnesting and unpivoting data stored in the SUPER data type.
- Improves performance of refreshing materialized views with SUPER data types.
- Adds support for aggregating INTERVAL literals with the ANY_VALUE function.
- Streaming ingestion now supports the following new SQL command to purge streaming data: `DELETE FROM streaming_materialized_views WHERE <where filter clause>`.

- The DECODE function replaces a specific value with either another specific value or a default value, depending on the result of an equality condition. DECODE now requires the following three parameters:
  - expression
  - search
  - result
- Adds functionality to stored procedures to allow for catching data overflow data type conversion errors, and handling inside an exception-handling block.
- You'll now receive an error when querying row-level security or dynamic data masking-protected relations if you change enable_case_sensitive_identifier to be different from the session default setting. Additionally, the following configuration is blocked when row-level security or dynamic data masking policies are applied in your provisioned cluster or serverless namespace:

  `ALTER USER <current_user> SET case-sensitive identifier.`
- The MERGE command now supports a simplified syntax that only requires the target and source table. For more information, see MERGE in the *Amazon Redshift Database Developer Guide*.
- Adds support for attaching identical dynamic data masking policies to multiple users or roles with the same priority, or without specifying the priority.
- You can now specify a COLLATION when adding a new column through ALTER TABLE ADD COLUMN.
- Fixes issue that delays the enforcement of QMR rules on concurrency scaling clusters and Amazon Redshift Serverless.
- Amazon Redshift Federated Query has expanded pushdown support for timezone with timestamp on Amazon RDS for PostgreSQL and Amazon Aurora PostgreSQL.
- You can now use Amazon RDS for MySQL and Aurora MySQL database names starting with digits with federated queries.
- Adds the SYS_ANALYZE_HISTORY view, which contains record details for ANALYZE operations.
- Adds the SYS_ANALYZE_COMPRESSION_HISTORY view, which contains record details for compression analysis operations during COPY or ANALYZE COMPRESSION commands.
- Adds the SYS_SESSION_HISTORY view, which contains record details related to active, historical, and restarted sessions.
- Adds the SYS_TRANSACTION_HISTORY view, which contains record details related to transaction level analysis that provides the time spent on commit, datasha number of blocks committed, and isolation level.

- Adds the SVV_REDSHIFT_SCHEMA_QUOTA view, which contains records related to quotas and the current disk usage for each schema in a database.
- Adds the SYS_PROCEDURE_CALL view, which contains records related to stored procedure calls, including start time, end time, status of the stored procedure call, and call hierarchy for nested stored procedure calls.
- Adds the SYS_CROSS_REGION_DATASHARING_USAGE view, which contains records related to tracking cross-Region data sharing usage.
- Adds the SYS_PROCEDURE_MESSAGES view, which contains records related to tracking information about logged stored procedure messages.
- Adds the SYS_UDF_LOG view, which contains records related to tracking system log messages from user-defined function calls, errors, warnings, or traces when applicable.
- Adds the new columns IS_RECURSIVE, IS_NESTED, S3LIST_TIME, and GET_PARTITION_TIME to SYS_EXTERNAL_QUERY_DETAIL.
- Adds MaxRPU, a new compute cost control setting for Redshift Serverless. With MaxRPU, you can optionally specify an upper compute threshold to control data warehouse costs at different points in time by selecting the maximum compute level that Redshift Serverless can scale per workgroup.
- Corrects the output of the INTERVAL literal with numeric interval strings. For example, an interval that specifies as `INTERVAL '1' YEAR` now returns `1 YEAR` instead of "`00:00:00`. In addition, the output of the INTERVAL literal is truncated to the smallest INTERVAL component specified. For example, `INTERVAL '1 day 1 hour 1 minute 1.123 seconds' HOUR TO MINUTE` is truncated to `1 day 01:01:00`.

Amazon Redshift patch 177

Cluster versions in this patch:

- 1.0.57922 - Trailing track version – Released on October 12th, 2023
- 1.0.57799 - Amazon Redshift Serverless version – Released on October 10th, 2023
- 1.0.57798 - Current track version – Released on October 10th, 2023
- 1.0.57085 - Trailing track version – Released on September 26th, 2023
- 1.0.56899 - Amazon Redshift Serverless version – Released on September 21st, 2023
- 1.0.56754 - Current track version – Released on September 21st, 2023
- 1.0.56242 - Current track version – Released on September 11th, 2023

- 1.0.55539 - Amazon Redshift Serverless version – Released on August 28th, 2023
- 1.0.55524 - Current track version – Released on August 28th, 2023
- 1.0.54899 - Current track version – Released on August 15th, 2023
- 1.0.54899 - Current track version – Released on August 14th, 2023
- 1.0.54899 - Current track version – Released on August 15th, 2023
- 1.0.54239 - Current track version – Released on August 3rd, 2023
- 1.0.54321 - Amazon Redshift Serverless version – Released on August 3rd, 2023

New features and improvements in this patch

- Adds the SYS_MV_STATE view, which contains a row for every state transition of a materialized view. SYS_MV_STATE can be used for MV refresh monitoring for Amazon Redshift Serverless and Amazon Redshift provisioned instances.
- Adds the SYS_USERLOG view, which records details for the changes to a database user for Create user, Drop user, Alter user (rename), Alter user (alter properties).
- Adds the SYS_COPY_REPLACEMENTS view, which displays a log that records when invalid UTF-8 characters were replaced by the COPY command with the ACCEPTINVCHARS option.
- Adds the SYS_SPATIAL_SIMPLIFY view, which contains information about simplified spatial geometry objects using the COPY command.
- Adds the SYS_VACUUM_HISTORY view, which you can use to see the details and results of VACUUM operations.
- Adds the SYS_SCHEMA_QUOTA_VIOLATIONS view to record the occurrence, timestamp, XID, and other useful information when a schema quota is exceeded.
- Adds the SYS_RESTORE_STATE view, which you can use monitor the redistribution progress of each table in the cluster during asynchronous classic resize.
- Adds the SYS_EXTERNAL_QUERY_ERROR view that return information about Redshift Spectrum scan errors.
- Adds the tag parameter to the CREATE MODEL command, so you can now track training costs with autopilot training jobs.
- Adds custom domain names (CNAME) for Amazon Redshift clusters.
- Adds preview support for Apache Iceberg, enabling customers to run analytics queries on Apache Iceberg tables within Amazon Redshift.
- Adds support for using user roles with parameter groups in workload management (WLM).

- Adds support for automatic mounting of AWS Glue Data Catalog, making it easier for customers to run queries in their data lakes.
- Adds functionality such that using grouping functions without a GROUP BY clause or using grouping operations in a WHERE clause results in an error.
- Adds functionality to stored procedures to allow for catching divide by zero errors and handling inside a exception-handling block.
- Fixes a bug that prevented queries from using concurrency scaling to write data to tables when the source table is a data sharing table.
- Fixes the case-sensitive identifier documented at enable_case_sensitive_identifier to now work with MERGE statements.
- Fixes the bug that a query on the function pg_get_late_binding_view_cols() might get ignored occasionally. You can now always cancel such queries.
- Improves performance for data sharing queries running on consumers when running vacuum jobs on the producer.
- Improves performance for data sharing and concurrency scaling queries, especially with concurrent data changes on the producer or when offloading to a concurrency scaling instance attached to the consumer.

Amazon Redshift patch 176

Cluster versions in this patch:

- 1.0.56738 - Trailing track version – released on September 21st, 2023
- 1.0.55837 - Trailing track version – released on September 11th, 2023
- 1.0.54776 - Current track version – released on August 15th, 2023
- 1.0.54052 - Current track version – Released on July 26th, 2023
- 1.0.53642 - Amazon Redshift Serverless version – Released on July 20th, 2023
- 1.0.53301 - Current track version – Released on July 20th, 2023
- 1.0.52943 - Amazon Redshift Serverless version – Released on July 7, 2023
- 1.0.52931 - Current track version – Released on July 7, 2023
- 1.0.52194 - Amazon Redshift Serverless version – Released on June 21, 2023
- 1.0.51986 - Current track version – Released on June 16, 2023
- 1.0.51594 - Current track version – Released on June 9, 2023

New features and improvements in this patch

- Improved error handling when writing GROUP BY () for an empty grouping set. This was ignored previously and now returns a parser error.
- Performance enhancements for incrementally refreshing materialized views with SUPER columns.
- ALTER TABLE <target_tbl> APPEND FROM <streaming_mv> – (ATA) SQL command now supports moving all records from a streaming materialized view (MV) as a source, in addition to tables as a source, to a target table. The support for ATA on streaming MVs allows users to rapidly purge all records in a streaming MV by moving them to another table to manage data growth.
- TRUNCATE <streaming_mv> – SQL command now supports truncating all records in a streaming materialized view (MV), in addition to tables. TRUNCATE deletes all records in the streaming MV, while leaving the streaming MV structure intact. Running TRUNCATE on streaming MVs allows customers to rapidly purge all records in a streaming MV to manage data growth.
- Added functionality for the QUALIFY clause to the SELECT command.
- Redshift machine-learning support for time series forecasting by integrating with Amazon Forecast.
- AWS Glue Data Catalog auto mounting is supported to simplify querying a data lake without extra steps to create external schema references.
- Altering an RLS policy is now supported. Refer to the documentation for more details at ALTER RLS POLICY.
- Lambda UDFs now support the STABLE function-volatility parameter in the CREATE FUNCTION statement. When the STABLE parameter is used in the CREATE FUNCTION statement and the Lambda UDF is called multiple times, with the same arguments, the expected number of Lambda UDF function invocations is decreased. The STABLE function volatility category is explained in more detail in the CREATE FUNCTION parameters.
- Multiple Lambda UDF performance improvements. Specifically, improved record batching support when querying a table protected by a row-level security (RLS) policy.
- Reduction in the overall encryption time for Amazon Redshift RA3 clusters and improvement in the availability of the data warehouse during encryption. For more information, see Amazon Redshift database encryption.
- A new system view SYS_MV_REFRESH_HISTORY has been added to Redshift. The SYS_MV_REFRESH_HISTORY view contains a row for the refresh activity of materialized views. Using SYS_MV_REFRESH_HISTORY, you can check the refresh history of materialized views.

SYS_MV_REFRESH_HISTORY is visible to all users. Superusers can see all rows; regular users can see only their own data.

A new column SPILLED_BLOCK_LOCAL_DISK has been added to system view SYS_QUERY_DETAIL. The new column SPILLED_BLOCK_LOCAL_DISK helps customers to determine blocks spilled to local disk. You can use SYS_QUERY_DETAIL to view details for queries at a step level. SYS_QUERY_DETAIL is visible to all users. Superusers can see all rows; regular users can see only metadata to which they have access.

- A new system view, SYS_QUERY_TEXT, has been added to Amazon Redshift Serverless and Amazon Redshift provisioned. The SYS_QUERY_TEXT view is similar to SVL_STATEMENTTEXT for provisioned clusters. Use the sequence column in the SYS_QUERY_TEXT view to get complete SQL statement text.

Amazon Redshift patch 175

Cluster versions in this patch:

- 1.0.53064 - Current track version – Released on July 7, 2023
- 1.0.51973 - Current track version – Released on June 16, 2023
- 1.0.51781 - Current track version – Released on June 10, 2023
- 1.0.51314 - Amazon Redshift Serverless version – Released on June 3, 2023
- 1.0.51304 - Current track version – Released on June 2, 2023
- 1.0.50708 - Current Track version – Released on May 19, 2023
- 1.0.50300 - Current track version – Released on May 8, 2023
- 1.0.49710 - Amazon Redshift Serverless version – Released on April 28, 2023
- 1.0.49676 - Current track version – Released on April 28, 2023

New features and improvements in this patch

- Minor bug fixes.
- Amazon Redshift streaming ingestion now supports cross-region streaming ingestion where your source Amazon Kinesis Data Streams (KDS) or Amazon Managed Streaming for Apache Kafka (MSK) topic can be located in an AWS region that is different from the AWS region where your Amazon Redshift data warehouse is located. The documentation at Getting started with

streaming ingestion from Amazon Kinesis Data Streams has been revised and explains how the REGION keyword is used.

- Egypt daylight saving adjustment.
- Improved overall times for encryption of RA3 clusters.

Amazon Redshift patch 174

1.0.51296 – Released on June 2, 2023

Release to trailing track. No release notes.

1.0.50468 – Released on May 12, 2023

Maintenance release. No release notes.

1.0.49780, 1.0.49868, and 1.0.49997 – Released on April 28, 2023

Release notes for this version:

- Improved batching support for Lambda UDF.
- Incremental batching for Lambda UDF.
- New MERGE SQL command to apply source data changes to Amazon Redshift tables.
- New dynamic data masking capability to simplify the process of protecting sensitive data in an Amazon Redshift data warehouse.
- New centralized access control for data sharing with Lake Formation that allows managing permission grants, viewing access controls, and auditing permissions on the tables and views in the Amazon Redshift datashares using Lake Formation APIs and the AWS Console.
- Egypt daylight saving adjustment.

1.0.49087 – Released on April 12, 2023

Maintenance release. No release notes.

1.0.48805 – Released on April 5, 2023

Release notes for this version:

- Amazon Redshift introduced additional performance enhancements to string-heavy queries using BYTEDICT, a new compression encoding in Amazon Redshift that speeds up string-based data processing between 5x to 63x compared to alternative compression encodings such as LZO or ZSTD. For more information on this feature, see Compression encodings in the *Amazon Redshift Database Developer Guide*.

1.0.48004 – Released on March 17, 2023

Maintenance release. No release notes.

1.0.47470 – Released on March 11, 2023

Release notes for this version:

- Improves query performance on `pg_catalog.svv_table_info`. Also adds new column `create_time`. When creating a table, this column stores the date/time stamp in UTC.
- Adds support for specifying session level timeout on federated query.

Amazon Redshift patch 173

1.0.49788 – Released on April 28, 2023

Release notes for this version:

- Egypt daylight saving adjustment.

1.0.49074 – Released on April 12, 2023

Release notes for this version:

- Timezone configuration updated to IANA library release 2022g.

1.0.48766 – Released on April 5, 2023

Maintenance release. No release notes.

1.0.48714 – Released on April 5, 2023

Maintenance release. No release notes.

1.0.48022 – Released on March 17, 2023

Maintenance release. No release notes.

1.0.47357 – Released on March 7, 2023

Maintenance release. No release notes.

1.0.46987 – Released on February 24, 2023

Maintenance release. No release notes.

1.0.46806 – Released on February 18, 2023

Maintenance release. No release notes.

1.0.46607 – Released on February 13, 2023

Release notes for this version:

- We now automatically convert tables with manually set interleaved sort keys to compound sort keys if their distribution style has been set to **DISTSTYLE KEY**, to improve the performance of these tables. This is done at the time of restoring a snapshot into Amazon Redshift Serverless.

1.0.45698 – Released on January 20, 2023

Release notes for this version:

- Adds a file extension parameter to the UNLOAD command, so file extensions are automatically added to filenames.
- Supports protecting RLS-protected objects by default when adding them to a datashare or if they're already part of a datashare. Administrators can now turn off RLS for datashares to allow consumers access to the protected object.
- Adds new system tables for monitoring: `SVV_ML_MODEL_INFO`, `SVV_MV_DEPENDENCY`, and `SYS_LOAD_DETAIL`. Also adds the columns `data_skewness` and `time_skewness` to the system table `SYS_QUERY_DETAIL`.

Amazon Redshift patch 172

Cluster versions in this patch:

- 1.0.46534 – Released on February 18, 2023
- 1.0.46523 – Released on February 13, 2023
- 1.0.46206 – Released on February 1, 2023
- 1.0.45603 – Released on January 20, 2023
- 1.0.44924 – Released on December 19, 2022
- 1.0.44903 – Released on December 18, 2022
- 1.0.44540 – Released on December 13, 2022
- 1.0.44126 – Released on November 23, 2022
- 1.0.43980 – Released on November 17, 2022

New features and improvements in this patch

- Tables created by CTAS are AUTO by default.
- Adds support for row-level security (RLS) on materialized views.
- Increases the S3 timeout to improve cross-Region data sharing.
- Adds new spatial function `ST_GeomFromGeohash`.
- Improves automatic selection of distribution key from composite primary keys to improve out-of-the-box performance.
- Adds automatic primary key to distribution key for tables with composite primary keys, improving out-of-the-box performance.
- Improves concurrency scaling to allow more queries to scale even as data changes.
- Improves data sharing query performance.
- Adds Machine Learning probability metrics for classification models.
- Adds new system tables for monitoring: `SVV_USER_INFO`, `SVV_MV_INFO`, `SYS_CONNECTION_LOG`, `SYS_DATASHARE_USAGE_PRODUCER`, `SYS_DATASHARE_USAGE_CONSUMER`, and `SYS_DATASHARE_CHANGE_LOG`.
- Adds support for querying VARBYTE columns in external tables for Parquet and ORC file types.

Amazon Redshift patch 171

Cluster versions in this patch:

- 1.0.43931 – Released on November 16, 2022
- 1.0.43551 – Released on November 5, 2022
- 1.0.43331 – Released on September 29, 2022
- 1.0.43029 – Released on September 26, 2022

New features and improvements in this patch

- CONNECT BY support: Adds support for the CONNECT BY SQL construct, letting you recursively query the hierarchical data in your data warehouse based on parent-child relationship within that data set.

Amazon Redshift patch 170

Cluster versions in this patch:

- 1.0.43922 – Released on November 21, 2022
- 1.0.43573 – Released on November 7, 2022
- 1.0.41881 – Released on September 20, 2022
- 1.0.41465 – Released on September 7, 2022
- 1.0.40325 – Released on July 27, 2022

New features and improvements in this patch

- ST_GeomfromGeoJSON: Constructs an Amazon Redshift spatial geometry object from VARCHAR in GeoJSON representation.

Amazon Redshift patch 169

Cluster versions in this patch:

- 1.0.41050 – Released on September 7, 2022

- 1.0.40083 – Released on July 16, 2022
- 1.0.39734 – Released on July 7, 2022
- 1.0.39380 – Released on June 23, 2022
- 1.0.39251 – Released on June 15, 2022
- 1.0.39009 – Released on June 8, 2022

New features and improvements in this patch

- Adds role as a parameter for the Alter Default Privileges command to support Role-based Access Control.
- Adds ACCEPTINVCHARS parameter to support replacing invalid UTF-8 characters when copying from Parquet and ORC files.
- Adds OBJECT(k,v) function to construct SUPER objects from key and value pairs.

Amazon Redshift patch 168

Cluster versions in this patch:

- 1.0.38698 – Released on May 25, 2022
- 1.0.38551 – Released on May 20, 2022
- 1.0.38463 – Released on May 18, 2022
- 1.0.38361 – Released on May 13, 2022
- 1.0.38199 – Released on May 9, 2022
- 1.0.38112 – Released on May 6, 2022
- 1.0.37684 – Released on April 20, 2022

New features and improvements in this patch

- Adds support for the Linear Learner model type in Amazon Redshift ML.
- Adds SNAPSHOT option for SQL transaction isolation level.
- Adds `farmhashFingerprint64` as new hashing algorithm for `VARBYTE` and `VARCHAR` data.
- Supports the AVG function in incremental refresh of materialized views.

- Supports correlated sub-queries on external tables in Redshift Spectrum.
- To improve the out-of-the-box query performance, Amazon Redshift automatically chooses a single column primary key for specific tables as a distribution key.

Behavior changes in Amazon Redshift

As Amazon Redshift continues to evolve and improve, certain changes in behavior are introduced to enhance performance, security, and user experience. This page serves as a comprehensive resource for you to stay informed about these critical updates, take actions, and avoid potential disruptions to your workloads.

Upcoming behavior changes

The following describes upcoming behavior changes.

Topics

- SUPER data type supports larger individual strings starting with Patch 206
- AWS KMS key permission enforcement for Amazon Redshift Serverless APIs after August 17, 2026
- Amazon Redshift enforces user lockout after multiple failed login attempts starting with Patch 208
- Enhanced billing model for manual snapshots on Amazon Redshift Serverless and Amazon Redshift RG instances effective June 08, 2026
- Iceberg DELETE on Lake Formation tables requires DELETE permission starting with Patch 202
- Amazon Redshift Serverless preserves zero-ETL and S3 event integrations on snapshot restore starting with Patch 202
- End of support for the Amazon Redshift ODBC 1.x driver on December 31, 2026
- Scalar Python UDFs will reach end of support after June 30, 2026
- Materialized View (MV) Auto-REFRESH Behavior Change after February 27, 2026
- Amazon Redshift won't support functions that access consumer information through datasharing after February 16, 2026
- Minimum Transport Layer Security (TLS) version changes effective starting October 31, 2026
- Amazon Redshift won't support the creation of new scalar Python UDFs after October 30, 2025

SUPER data type supports larger individual strings starting with Patch 206

Starting with Patch 206, Amazon Redshift supports individual strings up to 16,000,000 bytes within the SUPER data type (used to store semistructured data). This limit also applies to in-memory string operations. To use this capability, create a database parameter group and set the `enable_large_strings_opt_in` parameter to `Yes`. Then, attach the parameter group to your data warehouse. For provisioned clusters, you must reboot the cluster to apply the parameter change.

This change might affect you if you use SUPER data types to store large JSON documents. It might also affect you if your workloads rely on `VARCHAR(MAX)` in views, user-defined functions (UDFs), or stored procedures that process strings longer than 65,535 bytes.

When you set `enable_large_strings_opt_in` to `Yes`, in-memory operations resolve `VARCHAR(MAX)` to `VARCHAR(16000000)`. These operations include `CAST`, views, UDFs, and stored procedures. They can then process strings longer than 65,535 bytes. This change doesn't affect persistent storage for VARCHAR columns. `CREATE TABLE` with `VARCHAR(MAX)` continues to create `VARCHAR(65535)` columns. When you set the parameter to `No`, Amazon Redshift preserves the existing behavior of your data warehouse. To check the current state of your data warehouse, run `SELECT is_large_string_enabled();`.

When you set `enable_large_strings_opt_in` to `Yes`:

- `COPY` and `INSERT` into SUPER columns support sizes larger than the previous 65,535-byte limit. String functions such as `JSON_PARSE`, `JSON_SERIALIZE`, and `LISTAGG` also support these larger sizes.
- Existing views, UDFs, and stored procedures remain capped at 65,535 bytes until you recreate them.
- If an existing view uses `VARCHAR(MAX)` and other objects (such as other views) depend on it, `CREATE OR REPLACE VIEW` fails. The error is "cannot change data type of view column." To update the view, drop the dependent objects, drop and recreate the view, then recreate the dependent objects. You can replace views that use `VARCHAR(MAX)` and have no dependent objects without these extra steps.
- Keep this setting aligned between the data sharing producer and consumer data warehouses.

When you set `enable_large_strings_opt_in` to `No`:

- You can't `COPY` or `INSERT` strings longer than 65,535 bytes into SUPER columns. You can still read existing strings through SUPER. However, casting to `VARCHAR(MAX)` truncates the value to 65,535 bytes.
- String functions such as `JSON_PARSE`, `JSON_SERIALIZE`, and `LISTAGG` can't process VARCHAR values longer than 65,535 bytes.
- You can't query views, stored procedures, and UDFs that process string literals longer than 65,535 bytes.
- If you created a view with `VARCHAR(MAX)` while the parameter was set to `Yes`, and other objects (such as other views) depend on it, `CREATE OR REPLACE VIEW` fails with the same "cannot change data type of view column" error. Follow the same remediation steps described in the preceding list.
- A cross-instance snapshot restore can fail to ingest long strings. This failure occurs when the snapshot contains strings longer than 65,535 bytes in a SUPER column. It happens when you restore that snapshot to a cluster where this parameter is set to `No`.

For more information about patch versions, see Cluster versions for Amazon Redshift.

AWS KMS key permission enforcement for Amazon Redshift Serverless APIs after August 17, 2026

Amazon Redshift Serverless now performs a new authorization check that requires the IAM principal calling certain Amazon Redshift Serverless APIs to have explicit AWS KMS permissions on the customer managed key that encrypts the target namespace. Previously, Amazon Redshift performed AWS KMS operations on your behalf using the Amazon Redshift service role without verifying whether the calling IAM principal was itself authorized to use that key.

After the new authorization check is enabled, API calls from principals that lack the required AWS KMS permissions will fail with a `ValidationException: "Insufficient KMS permissions."` You may be impacted by this if you have Amazon Redshift Serverless namespaces encrypted with a customer managed key and use IAM principals to call the affected APIs listed below.

To avoid disruption, review which IAM principals in your accounts call the APIs listed below against namespaces encrypted with a customer managed key, and ensure those principals have the required AWS KMS permissions on the key. You can grant these permissions through an AWS KMS key policy, an IAM identity-based policy, or an AWS KMS grant.

This change does not affect namespaces encrypted with the default AWS-owned key. If your IAM principals already have the required AWS KMS permissions listed below, no action is needed.

The affected APIs and required permissions on the namespace's customer managed key are listed below:

- `CreateWorkgroup`, `RestoreFromSnapshot`, `RestoreFromRecoveryPoint` – require `kms:Encrypt`, `kms:GenerateDataKey`, `kms:CreateGrant`, and `kms:Decrypt`.
- `RestoreTableFromSnapshot`, `RestoreTableFromRecoveryPoint` – require `kms:Decrypt`.

For more information, see Controlling access to AWS KMS keys in the *AWS Key Management Service Developer Guide*. For more information about Amazon Redshift database encryption, see Amazon Redshift database encryption.

Amazon Redshift enforces user lockout after multiple failed login attempts starting with Patch 208

Starting with Amazon Redshift Patch 208, to enhance the security posture of your data warehouse, Amazon Redshift automatically locks a database user after it exceeds the configured number of consecutive failed login attempts. The feature is enabled by default with a threshold of 5 failed attempts (configurable from 2 to 50) and applies to both provisioned clusters and serverless workgroups.

You may be impacted by this if you connect using database (password-based) connections. Once locked, login attempts are rejected even with the correct password until an administrator or superuser unlocks it. Federated identities that authenticate through IAM or AWS IAM Identity Center are not affected, because they use short-lived tokens rather than static passwords.

To manage this behavior:

- Set the threshold with `ALTER SYSTEM SET max_failed_login_attempts TO value;` (2–50, default 5).
- Check lockout status for all users with `SHOW USER LOCKOUT;`.
- Unlock user with `ALTER USER username LOGIN PASSWORD 'password';`, which also resets the failed-attempt counter.

For more information, see SHOW USER LOCKOUT, ALTER USER, and max_failed_login_attempts.

Enhanced billing model for manual snapshots on Amazon Redshift Serverless and Amazon Redshift RG instances effective June 08, 2026

Starting June 08, 2026, Amazon Redshift announces an enhanced billing model for manual snapshots on Amazon Redshift Serverless and Amazon Redshift RG instances. With this enhancement, manual snapshots are now metered based on unique data blocks stored across all active manual snapshots in your account. Shared data blocks that appear in multiple snapshots are counted only once, billed at existing manual snapshot rates for your region.

You may be impacted by this if you use manual snapshots on Amazon Redshift Serverless workgroups or Amazon Redshift RG clusters. This results in lower manual snapshot costs for customers who maintain multiple snapshots.

No action is required. The enhanced billing model automatically applies to both existing and new manual snapshots.

For more information about snapshot pricing, see Amazon Redshift pricing.

Iceberg DELETE on Lake Formation tables requires DELETE permission starting with Patch 202

Starting with Amazon Redshift Patch 202, Iceberg DELETE operations on Lake Formation (LF) managed tables require the DELETE Lake Formation permission. UPDATE and MERGE operations require both INSERT and DELETE permissions. All Iceberg DML operations require ALTER permission.

You may be impacted by this if you perform DELETE, UPDATE, or MERGE operations against Iceberg tables managed by AWS Lake Formation.

Previously, principals with only INSERT permission (and no DELETE permission) could perform delete operations on LF-managed Iceberg tables. Starting with Patch 202, DELETE operations require the DELETE Lake Formation permission. Principals relying solely on INSERT permission to perform deletes will receive a permission error until DELETE permission is explicitly granted. S3 Tables are not affected by this change.

To continue performing DELETE operations after this change, review your Lake Formation permission grants for Iceberg tables and ensure that principals performing delete operations have the DELETE Lake Formation permission granted. You can verify existing grants using the AWS Lake Formation console or the `aws lakeformation list-permissions` AWS CLI command.

If you need to temporarily revert to the previous behavior, contact AWS Support to disable this change on your cluster without requiring a code deployment.

For more information about patch versions, see Cluster versions for Amazon Redshift.

Amazon Redshift Serverless preserves zero-ETL and S3 event integrations on snapshot restore starting with Patch 202

Starting with Amazon Redshift Patch 202, when you restore an Amazon Redshift Serverless namespace from a snapshot or recovery point to the same serverless namespace, zero-ETL and S3 event integrations associated with the namespace and snapshot/recovery point are maintained automatically. No additional action is required.

You may be impacted by this if you restore a snapshot or recovery point to Amazon Redshift Serverless namespaces that have zero-ETL or S3 event integrations configured.

Previously, restoring a snapshot or recovery point to a serverless namespace marked associated zero-ETL and S3 event integrations as failed, requiring you to manually recreate them after the restore completed. Starting with Patch 202, these integrations are maintained by default and resume operating after the restore completes.

This feature applies to Amazon Redshift Serverless only when restored to the same serverless namespace. Restoring a snapshot to a different namespace does not maintain integrations. Snapshot restores on provisioned clusters do not maintain zero-ETL or S3 event integrations.

To opt out of maintaining integrations during a restore, uncheck the **Maintain Integrations** box on the restore page in the AWS Management Console. If you are using the AWS CLI, set the `--no-maintain-integration` parameter when calling the restore-from-snapshot or restore-from-recovery-point API operation. When you opt out, integrations enter the `FAILED` state after the restore. You can then delete the failed integrations and recreate them.

For more information about patch versions, see Cluster versions for Amazon Redshift.

End of support for the Amazon Redshift ODBC 1.x driver on December 31, 2026

Beginning December 31, 2026, Amazon Redshift will discontinue support for the ODBC 1.x driver. Based on customer feedback, we have extended the original end-of-support date from September

30, 2026 to December 31, 2026, to provide additional time for migration. This applies to both Amazon Redshift provisioned clusters and serverless workgroups.

You may be impacted by this if you use any version of the ODBC 1.x driver to connect to Amazon Redshift. To verify whether you are using an ODBC 1.x driver, run the following query:

```
SELECT *
FROM SYS_CONNECTION_LOG
WHERE
(driver_version ilike 'Amazon Redshift ODBC Driver 1%'
OR driver_version ilike 'Redshift ODBC Driver 01%'
OR driver_version ilike 'Redshift ODBC Driver 1,%'
)
OR
(application_name ilike 'Amazon Redshift ODBC Driver 1%');
```

To continue receiving technical support for your Amazon Redshift ODBC driver connections, please migrate to the latest Amazon Redshift ODBC 2.x driver before December 30, 2026.

Before migrating to the ODBC 2.x driver in a production environment, we recommend conducting a thorough proof of concept to verify that the new driver meets all your functional requirements.

We recommend using the latest version of the Amazon Redshift ODBC 2.x driver and setting the ApplicationName property to identify your application when connecting to Amazon Redshift.

Scalar Python UDFs will reach end of support after June 30, 2026

Amazon Redshift will end support for Python UDFs after June 30, 2026. As an alternative, we recommend that you use Lambda UDFs.

Lambda UDFs have the following advantages over Python UDFs:

- Lambda UDFs can connect to external services and APIs from within the UDF logic.
- Lambda UDFs use Lambda compute resources. Heavy compute- or memory-intensive Lambda UDFs don't impact Amazon Redshift's query performance or resource concurrency.
- Lambda UDFs support running Python code. Lambda UDFs support multiple Python runtimes depending on specific use cases. For more information, see Building with Python in the *AWS Lambda Developer Guide*.
- You can isolate custom code execution in a separate service boundary. This simplifies maintenance, monitoring, budgeting, and permission management.

For information on creating and using Lambda UDFs, see Scalar Lambda UDFs in the *Amazon Redshift Database Developer Guide*. For information on converting existing Python UDFs to Lambda UDFs, see the blog post .

Materialized View (MV) Auto-REFRESH Behavior Change after February 27, 2026

Starting February 27, 2026, Auto REFRESH queries for Amazon Redshift materialized views are executed as user queries rather than background autonomic processes. As a result, Auto REFRESH queries now run with the same priority as other user queries.

This change improves the freshness of materialized views with Auto REFRESH enabled, helping them stay more up to date with the latest changes to their base tables compared to the previous behavior.

Note: The MV Auto REFRESH behavior change feature is only enabled for Amazon Redshift Provisioned clusters on the CURRENT Track of patch release P198 and newer. It is currently disabled on Serverless.

Amazon Redshift won't support functions that access consumer information through datasharing after February 16, 2026

Starting February 16, 2026, Amazon Redshift will no longer support the usage of `user_is_member_of` and related functions that access consumer user, role, or group information through datasharing.

Minimum Transport Layer Security (TLS) version changes effective starting October 31, 2026

Beginning October 31, 2026, Amazon Redshift will enforce a minimum Transport Layer Security (TLS) version of 1.2. Incoming connections that use TLS versions 1.0 or 1.1 will be rejected. This applies to both Amazon Redshift provisioned clusters and serverless workgroups. Amazon Redshift data warehouses not using TLS will not be affected by this change.

This update might impact you if you use TLS versions 1.0 or 1.1 to connect to Amazon Redshift.

To verify which TLS version you are currently using, you can:

For Amazon Redshift Provisioned: Check the sslversion column in the STL_CONNECTION_LOG system table [1].

For Amazon Redshift Serverless Workgroup: Check the ssl_version column in the SYS_CONNECTION_LOG system table [2].

To maintain uninterrupted access to your Amazon Redshift data warehouse after this change, please follow the steps listed below:

1. Update your client to support TLS 1.2 or higher
2. Install the latest driver version with TLS 1.2+ support

We recommend using the latest version of the Amazon Redshift driver [3] if possible.

[1] https://docs.aws.amazon.com/redshift/latest/dg/r_STL_CONNECTION_LOG.html

[2] https://docs.aws.amazon.com/redshift/latest/dg/SYS_CONNECTION_LOG.html

[3] https://docs.aws.amazon.com/redshift/latest/mgmt/configuring-connections.html

Amazon Redshift won't support the creation of new scalar Python UDFs after October 30, 2025

Amazon Redshift will no longer support the creation of new Python UDFs after October 30, 2025. Existing Python UDFs will continue to function normally. We strongly recommend that you migrate your existing Python UDFs to Lambda UDFs before this date.

Lambda UDFs have the following advantages over Python UDFs:

- Lambda UDFs can connect to external services and APIs from within the UDF logic.
- Lambda UDFs use Lambda compute resources. Heavy compute- or memory-intensive Lambda UDFs don't impact Amazon Redshift's query performance or resource concurrency.
- Lambda UDFs support running Python code. Lambda UDFs support multiple Python runtimes depending on specific use cases. For more information, see Building with Python in the *AWS Lambda Developer Guide*.
- You can isolate custom code execution in a separate service boundary. This simplifies maintenance, monitoring, budgeting, and permission management.

For information on creating and using Lambda UDFs, see Scalar Lambda UDFs in the *Amazon Redshift Database Developer Guide*. For information on converting existing Python UDFs to Lambda UDFs, see the blog post .

Recent behavior changes

Topics

- Amazon Redshift takes automated snapshots of paused data sharing producer clusters starting September 30, 2026
- Amazon Redshift uses up-to-date IANA Time Zone Database after Aug 26, 2025
- Amazon Redshift Serverless RPU changes effective after August 15, 2025
- Database audit logging changes effective after August 10, 2025
- Virtual Private Cloud Endpoint changes for serverless workgroups effective after June 27, 2025
- Query monitoring changes effective after May 2, 2025
- Security changes effective after January 10, 2025

Amazon Redshift takes automated snapshots of paused data sharing producer clusters starting September 30, 2026

Starting September 30, 2026, Amazon Redshift takes automated snapshots of a paused data sharing producer cluster. Because consumers can write to a paused producer, these snapshots capture those data sharing writes and provide a recovery point.

You may be impacted by this if you pause data sharing producer clusters.

Previously, Amazon Redshift didn't take automated snapshots of these clusters while they were paused, so the most recent automated snapshot available for restore was the one taken when the cluster was paused. Automated snapshots now continue to be created throughout the paused period.

These snapshots follow the cluster's existing automated snapshot schedule and retention period, and no action is required. For a paused cluster that isn't a data sharing producer, Amazon Redshift doesn't take automated snapshots while the cluster is paused.

For more information, see Automated snapshots.

Amazon Redshift uses up-to-date IANA Time Zone Database after Aug 26, 2025

Beginning Aug 26, 2025, Amazon Redshift calculates time zones by adopting the latest IANA Time Zone Database patches. This change alters how date and time conversions function for

certain time zones and time periods. This update affects explicit time zone conversions, such as those performed with the CONVERT_TIMEZONE function or the TIMEZONE and AT TIME ZONE commands, as well as implicit conversions that occur during type casting operations, particularly between TIMESTAMP and TIMESTAMPTZ formats.

The following is a list of updates for time zone and time period combinations:

- Time zones now correctly observe daylight saving time (DST) after year 2038. Previously, no time zone observed DST after 2038.
- The `America/Toronto` time zone, and the time zones that link to it, have DST switch in 1947-1950 at 2 AM local time instead of at midnight.
- Amazon Redshift now correctly reflects local mean time (LMT) for periods before standardization for all time zones. This period is specific to a time zone, with most time zones switching to standardization before mid-19th century.
- `EET`, `CET`, `WET` and `MET` are now treated as normal time zones instead of abbreviations.
- The following time zone names no longer exist in Amazon Redshift:
  - `Asia/Riyadh87`
  - `Asia/Riyadh88`
  - `Asia/Riyadh89`
  - `Mideast/Riyadh87`
  - `Mideast/Riyadh88`
  - `Mideast/Riyadh89`
  - `US/Pacific-New`

For more information on the IANA Time Zone Database, see Time Zone Database on the IANA Time Zone Database website.

Amazon Redshift Serverless RPU changes effective after August 15, 2025

Beginning August 15, 2025, the AWS account quota for Amazon Redshift Serverless base Redshift Processing Units (RPUs) is the greater of either 3,200 RPUs or 1.5 times your maximum aggregate base RPUs from the previous six months.

Database audit logging changes effective after August 10, 2025

Beginning August 10, 2025, Amazon Redshift is making a change to Database Audit Logging, which requires your action. Amazon Redshift logs information about connections and user activities in your database to Amazon S3 buckets and CloudWatch. After August 10, 2025, Amazon Redshift will discontinue database audit logging to your Amazon S3 buckets which have a bucket policy specifying a Redshift IAM USER. We recommend updating your policies to use the Redshift SERVICE-PRINCIPAL instead, within S3 bucket policies for audit logging. For information about audit logging, see Bucket permissions for Amazon Redshift audit logging.

To avoid any logging interruption, review and update your S3 bucket policies to grant access to the Redshift service-principal in the associated region prior to August 10, 2025. For information about database audit logging, see Log files in Amazon S3

For questions or concerns, contact AWS support at the following link: AWS Support.

Virtual Private Cloud Endpoint changes for serverless workgroups effective after June 27, 2025

Beginning June 27, 2025, Amazon Redshift is making a change to Virtual Private Cloud Endpoint (VPCE) support for serverless workgroups. Prior to this date, Amazon Redshift deploys with endpoints into a single Availability Zone (AZ) during workgroup creation, and expands VPCE support up to three AZs over time. After this date, Amazon Redshift deploys VPCEs in up to three of the Availability Zones specified during workgroup creation.

For more information, see Considerations when using Amazon Redshift Serverless.

For questions or concerns, contact AWS support at the following link: AWS Support.

Topics

- Query monitoring changes effective after May 2, 2025
- Security changes effective after January 10, 2025

Query monitoring changes effective after May 2, 2025

Effective May 2, 2025, we will no longer offer the Query CPU time (`max_query_cpu_time`) and Query CPU usage (`max_query_cpu_percentage`) metrics from the **Query Limits** tab for both

existing and newly created Redshift Serverless workgroups. After this date, we will automatically remove all query limits based on these metrics across all Redshift Serverless workgroups.

Query limits are designed to catch runaway queries. However, Query CPU time (`max_query_cpu_time`) and Query CPU usage (`max_query_cpu_percentage`) can vary during the lifetime of the query, and are thus not a consistently effective method to catch runaway queries. To catch runaway queries, we recommend that you leverage query monitoring metrics that provide consistent and actionable information. Some examples include:

- **Query execution time** (`max_query_execution_time`): To ensure queries complete within expected timeframes.
- **Return row count** (`max_scan_row_count`): To monitor the scale of data being processed.
- **Query queue time** (`max_query_queue_time`): To identify queries that spend time queueing.

For a full list of supported metrics, see Query monitoring metrics for Amazon Redshift Serverless.

Security changes effective after January 10, 2025

Security is our top priority at Amazon Web Services (AWS). To that end, we are further strengthening the security posture of Amazon Redshift environments by introducing enhanced security defaults which help you adhere to best practices in data security without requiring additional setup and reduce the risk of potential misconfiguration. To avoid any potential disruption, review your provisioned cluster and serverless workgroup creation configurations, scripts, and tools to make necessary changes to align with the new default settings before the effective date.

Public access disabled by default

After January 10, 2025, public accessibility will be disabled by default for all newly created provisioned clusters, and for clusters restored from snapshots. With this release, by default, connections to clusters will only be permitted from client applications within the same Virtual Private Cloud (VPC). To access your data warehouse from applications in another VPC, configure cross-VPC access. This change will be reflected in the `CreateCluster` and `RestoreFromClusterSnapshot` API operations, and corresponding SDK and AWS CLI commands. If you create a provisioned cluster from the Amazon Redshift console, then the cluster has public access disabled by default.

In case you still need public access, you will need to override the default and set the `PubliclyAccessible` parameter to true when you run `CreateCluster` or `RestoreFromClusterSnapshot` API operations. With a publicly accessible cluster, we recommend that you must use security groups or network access control lists (ACLs) to restrict access. For more information, see VPC security groups and Configuring security group communication settings for an Amazon Redshift cluster or an Amazon Redshift Serverless workgroup.

Encryption by default

After January 10, 2025, Amazon Redshift will further enhance data and cluster security by enabling encryption as the default setting for all newly created Amazon Redshift provisioned clusters. This does not apply to clusters restored from snapshots.

With this change, the ability to decrypt clusters will no longer be available when using the AWS Management Console, AWS CLI, or API to create a provisioned cluster without specifying a KMS key. The cluster will automatically be encrypted with an AWS owned key.

This update might impact you if you are creating unencrypted clusters using automated scripts or leveraging data sharing with unencrypted clusters. To ensure a seamless transition, update your scripts that create unencrypted clusters. Additionally, if you regularly create new unencrypted consumer clusters and use them for data sharing, review your configurations to ensure the producer and consumer clusters are both encrypted, preventing disruptions to your data sharing activities. For more information, see Amazon Redshift database encryption.

Enforcing SSL connections

After January 10, 2025, Amazon Redshift will enforce SSL connections by default for clients connecting to newly created provisioned and restored clusters. This default change will also apply to serverless workgroups.

With this change, a new default parameter group named `default.redshift-2.0` will be introduced for all newly created or restored clusters, with the `require_ssl` parameter set to `true` by default. Any new clusters created without a specified parameter group will automatically utilize the `default.redshift-2.0` parameter group. When creating a cluster through the Amazon Redshift console, the new `default.redshift-2.0` parameter group will be automatically selected. This change will also be reflected in the `CreateCluster` and `RestoreFromClusterSnapshot` API operations, and the corresponding SDK and AWS CLI commands. If you use existing or custom parameter groups, Amazon Redshift will continue to

honor the `require_ssl` value specified in your parameter group. You continue to have the option to change the `require_ssl` value in your custom parameter groups as needed.

For Amazon Redshift Serverless users, the default value of `require_ssl` in the `config-parameters` will be changed to `true`. Any requests to create new workgroups with `require_ssl` set to `false` will be rejected. You can change the `require_ssl` value to `false` after the workgroup is created. For more information, see Configuring security options for connections.

Note that you will still have the ability to modify cluster or workgroup settings to change the default behavior, if needed for your specific use cases.

Code examples for Amazon Redshift using AWS SDKs

The following code examples show how to use Amazon Redshift with an AWS software development kit (SDK).

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Code examples

- Basic examples for Amazon Redshift using AWS SDKs
  - Hello Amazon Redshift
  - Learn the basics of Amazon Redshift with an AWS SDK
  - Actions for Amazon Redshift using AWS SDKs
    - Use CreateCluster with an AWS SDK or CLI
    - Use DeleteCluster with an AWS SDK or CLI
    - Use DescribeClusters with an AWS SDK or CLI
    - Use DescribeStatement with an AWS SDK
    - Use ExecuteStatement with an AWS SDK
    - Use GetStatementResult with an AWS SDK
    - Use ListDatabases with an AWS SDK
    - Use ModifyCluster with an AWS SDK or CLI
- Scenarios for Amazon Redshift using AWS SDKs
  - Create an Amazon Redshift item tracker
  - Get started with serverless data warehouses using the CLI
  - Getting started with provisioned data warehouse clusters

Basic examples for Amazon Redshift using AWS SDKs

The following code examples show how to use the basics of Amazon Redshift with AWS SDKs.

Examples

- Hello Amazon Redshift
- Learn the basics of Amazon Redshift with an AWS SDK
- Actions for Amazon Redshift using AWS SDKs
  - Use CreateCluster with an AWS SDK or CLI
  - Use DeleteCluster with an AWS SDK or CLI
  - Use DescribeClusters with an AWS SDK or CLI
  - Use DescribeStatement with an AWS SDK
  - Use ExecuteStatement with an AWS SDK
  - Use GetStatementResult with an AWS SDK
  - Use ListDatabases with an AWS SDK
  - Use ModifyCluster with an AWS SDK or CLI

Hello Amazon Redshift

The following code examples show how to get started using Amazon Redshift.

.NET

SDK for .NET (v4)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/// <summary>
/// Main method to run the Hello Amazon Redshift example.
/// </summary>
/// <param name="args">Command line arguments (not used).</param>
```

```
   public static async Task Main(string[] args)
   {
       var redshiftClient = new AmazonRedshiftClient();

       Console.WriteLine("Hello, Amazon Redshift! Let's list available
clusters:");

       var clusters = new List<Cluster>();

       try
       {
           // Use pagination to retrieve all clusters.
           var clustersPaginator =
redshiftClient.Paginators.DescribeClusters(new DescribeClustersRequest());

           await foreach (var response in clustersPaginator.Responses)
           {
               if (response.Clusters != null)
                   clusters.AddRange(response.Clusters);
           }

           Console.WriteLine($"{clusters.Count} cluster(s) retrieved.");

           foreach (var cluster in clusters)
           {
               Console.WriteLine($"\t{cluster.ClusterIdentifier} (Status:
{cluster.ClusterStatus})");
           }
       }
       catch (AmazonRedshiftException ex)
       {
           Console.WriteLine($"Couldn't list clusters. Here's why:
{ex.Message}");
       }
       catch (Exception ex)
       {
           Console.WriteLine($"An error occurred: {ex.Message}");
       }
   }
```

- For API details, see DescribeClusters in *AWS SDK for .NET API Reference*.

Go

SDK for Go V2

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
)

// main uses the AWS SDK for Go V2 to create a Redshift client
// and list up to 10 clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 redshiftClient := redshift.NewFromConfig(sdkConfig)
 count := 20
 fmt.Printf("Let's list up to %v clusters for your account.\n", count)
 result, err := redshiftClient.DescribeClusters(ctx,
 &redshift.DescribeClustersInput{
  MaxRecords: aws.Int32(int32(count)),
 })
```

```
 if err != nil {
  fmt.Printf("Couldn't list clusters for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Clusters) == 0 {
  fmt.Println("You don't have any clusters!")
  return
 }
 for _, cluster := range result.Clusters {
  fmt.Printf("\t%v : %v\n", *cluster.ClusterIdentifier, *cluster.ClusterStatus)
 }
}
```

- For API details, see DescribeClusters in *AWS SDK for Go API Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import
 software.amazon.awssdk.services.redshift.paginators.DescribeClustersIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloRedshift {
```

```
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RedshiftClient redshiftClient = RedshiftClient.builder()
            .region(region)
            .build();

        listClustersPaginator(redshiftClient);
    }

    public static void listClustersPaginator(RedshiftClient redshiftClient) {
        DescribeClustersIterable clustersIterable =
 redshiftClient.describeClustersPaginator();
        clustersIterable.stream()
            .flatMap(r -> r.clusters().stream())
            .forEach(cluster -> System.out
                .println(" Cluster identifier: " + cluster.clusterIdentifier() +
 " status = " + cluster.clusterStatus()));
    }
}
```

- For API details, see DescribeClusters in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import boto3


def hello_redshift(redshift_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Redshift client and
 list
    the clusters in your account. This list might be empty if you haven't created
    any clusters.
```

```
    This example uses the default settings specified in your shared credentials
    and config files.

    :param redshift_client: A Boto3 Redshift Client object.
    """
    print("Hello, Redshift! Let's list your clusters:")
    paginator = redshift_client.get_paginator("describe_clusters")
    clusters = []
    for page in paginator.paginate():
        clusters.extend(page["Clusters"])

    print(f"{len(clusters)} cluster(s) were found.")

    for cluster in clusters:
        print(f"  {cluster['ClusterIdentifier']}")


if __name__ == "__main__":
    hello_redshift(boto3.client("redshift"))
```

- For API details, see DescribeClusters in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Learn the basics of Amazon Redshift with an AWS SDK

The following code examples show how to:

- Create a Redshift cluster.
- List databases in the cluster.
- Create a table named Movies.
- Populate the Movies table.
- Query the Movies table by year.
- Modify the Redshift cluster.
- Delete the Amazon Redshift cluster.

.NET

SDK for .NET (v4)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a Redshift wrapper class to manage operations.

```
/// <summary>
/// Wrapper class for Amazon Redshift operations.
/// </summary>
public class RedshiftWrapper
{
    private readonly IAmazonRedshift _redshiftClient;
    private readonly IAmazonRedshiftDataAPIService _redshiftDataClient;

    /// <summary>
    /// Constructor for RedshiftWrapper.
    /// </summary>
    /// <param name="redshiftClient">Amazon Redshift client.</param>
    /// <param name="redshiftDataClient">Amazon Redshift Data API client.</param>
    public RedshiftWrapper(IAmazonRedshift redshiftClient,
 IAmazonRedshiftDataAPIService redshiftDataClient)
    {
        _redshiftClient = redshiftClient;
        _redshiftDataClient = redshiftDataClient;
    }

    /// <summary>
    /// Create a new Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="databaseName">The name of the database.</param>
    /// <param name="masterUsername">The master username.</param>
    /// <param name="masterUserPassword">The master user password.</param>
    /// <param name="nodeType">The node type for the cluster.</param>
    /// <returns>The cluster that was created.</returns>
    public async Task<Cluster> CreateClusterAsync(string clusterIdentifier,
 string databaseName,
```

```
        string masterUsername, string masterUserPassword, string nodeType =
"ra3.large")
    {
        try
        {
            var request = new CreateClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DBName = databaseName,
                MasterUsername = masterUsername,
                MasterUserPassword = masterUserPassword,
                NodeType = nodeType,
                NumberOfNodes = 1,
                ClusterType = "single-node"
            };

            var response = await _redshiftClient.CreateClusterAsync(request);
            Console.WriteLine($"Created cluster {clusterIdentifier}");
            return response.Cluster;
        }
        catch (ClusterAlreadyExistsException ex)
        {
            Console.WriteLine($"Cluster already exists: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't create cluster. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Describe Amazon Redshift clusters.
    /// </summary>
    /// <param name="clusterIdentifier">Optional cluster identifier to describe a
specific cluster.</param>
    /// <returns>A list of clusters.</returns>
    public async Task<List<Cluster>> DescribeClustersAsync(string?
clusterIdentifier = null)
    {
        try
        {
```

```
            var clusters = new List<Cluster>();
            var request = new DescribeClustersRequest();
            if (!string.IsNullOrEmpty(clusterIdentifier))
            {
                request.ClusterIdentifier = clusterIdentifier;
            }

            var clustersPaginator =
_redshiftClient.Paginators.DescribeClusters(request);
            await foreach (var response in clustersPaginator.Responses)
            {
                if (response.Clusters != null)
                    clusters.AddRange(response.Clusters);
            }

            Console.WriteLine($"{clusters.Count} cluster(s) retrieved.");
            foreach (var cluster in clusters)
            {
                Console.WriteLine($"\t{cluster.ClusterIdentifier} (Status:
{cluster.ClusterStatus})");
            }

            return clusters;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
{ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe clusters. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Modify an Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="preferredMaintenanceWindow">The preferred maintenance
window.</param>
```

```
   /// <returns>True if successful.</returns>
   public async Task<bool> ModifyClusterAsync(string clusterIdentifier, string
preferredMaintenanceWindow)
   {
       try
       {
           var request = new ModifyClusterRequest
           {
               ClusterIdentifier = clusterIdentifier,
               PreferredMaintenanceWindow = preferredMaintenanceWindow
           };

           var response = await _redshiftClient.ModifyClusterAsync(request);
           Console.WriteLine($"The modified cluster was successfully modified
and has {response.Cluster.PreferredMaintenanceWindow} as the maintenance
window");
           return true;
       }
       catch (ClusterNotFoundException ex)
       {
           Console.WriteLine($"Cluster {clusterIdentifier} not found:
{ex.Message}");
           return false;
       }
       catch (Exception ex)
       {
           Console.WriteLine($"Couldn't modify cluster. Here's why:
{ex.Message}");
           return false;
       }
   }

   /// <summary>
   /// Delete an Amazon Redshift cluster without a final snapshot.
   /// </summary>
   /// <param name="clusterIdentifier">The identifier for the cluster.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> DeleteClusterWithoutSnapshotAsync(string
clusterIdentifier)
   {
       try
       {
           var request = new DeleteClusterRequest
           {
```

```
                ClusterIdentifier = clusterIdentifier,
                SkipFinalClusterSnapshot = true
            };

            var response = await _redshiftClient.DeleteClusterAsync(request);
            Console.WriteLine($"The {clusterIdentifier} was deleted");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster not found: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete cluster. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// List databases in a Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="dbUser">The database name for authentication.</param>
    /// <returns>A list of database names.</returns>
    public async Task<List<string>> ListDatabasesAsync(string clusterIdentifier,
string dbUser, string databaseName)
    {
        try
        {
            var request = new ListDatabasesRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DbUser = dbUser,
                Database = databaseName
            };

            var response = await _redshiftDataClient.ListDatabasesAsync(request);
            var databases = new List<string>();

            foreach (var database in response.Databases)
```

```
            {
                Console.WriteLine($"The database name is : {database}");
                databases.Add(database);
            }

            return databases;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list databases. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create a table in the Redshift database.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <returns>The statement ID.</returns>
    public async Task<string> CreateTableAsync(string clusterIdentifier, string
database, string dbUser)
    {
        try
        {
            var sqlStatement = @"
                CREATE TABLE Movies (
                    id INTEGER PRIMARY KEY,
                    title VARCHAR(250) NOT NULL,
                    year INTEGER NOT NULL
                )";

            var request = new ExecuteStatementRequest
            {
                ClusterIdentifier = clusterIdentifier,
                Database = database,
                DbUser = dbUser,
```

```
                Sql = sqlStatement
            };

            var response = await
_redshiftDataClient.ExecuteStatementAsync(request);
            await WaitForStatementToCompleteAsync(response.Id);
            Console.WriteLine("Table created: Movies");
            return response.Id;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't create table. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Insert a record into the Movies table using parameterized query.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="id">The movie ID.</param>
    /// <param name="title">The movie title.</param>
    /// <param name="year">The movie year.</param>
    /// <returns>The statement ID.</returns>
    public async Task<string> InsertMovieAsync(string clusterIdentifier, string
database, string dbUser,
        int id, string title, int year)
    {
        try
        {
            var sqlStatement = "INSERT INTO Movies (id, title, year) VALUES
(:id, :title, :year)";

            var request = new ExecuteStatementRequest
            {
                ClusterIdentifier = clusterIdentifier,
```

```
                Database = database,
                DbUser = dbUser,
                Sql = sqlStatement,
                Parameters = new List<SqlParameter>
                {
                    new SqlParameter { Name = "id", Value = id.ToString() },
                    new SqlParameter { Name = "title", Value = title },
                    new SqlParameter { Name = "year", Value = year.ToString() }
                }
            };

            var response = await
_redshiftDataClient.ExecuteStatementAsync(request);
            await WaitForStatementToCompleteAsync(response.Id);
            Console.WriteLine($"Inserted: {title} ({year})");
            return response.Id;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't insert movie. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Query movies by year using parameterized query.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="year">The year to query.</param>
    /// <returns>A list of movie titles.</returns>
    public async Task<List<string>> QueryMoviesByYearAsync(string
clusterIdentifier, string database,
        string dbUser, int year)
    {
        try
        {
```

```
            var sqlStatement = "SELECT title FROM Movies WHERE year = :year";

            var request = new ExecuteStatementRequest
            {
                ClusterIdentifier = clusterIdentifier,
                Database = database,
                DbUser = dbUser,
                Sql = sqlStatement,
                Parameters = new List<SqlParameter>
                {
                    new SqlParameter { Name = "year", Value = year.ToString() }
                }
            };

            var response = await
_redshiftDataClient.ExecuteStatementAsync(request);
            Console.WriteLine($"The identifier of the statement is
{response.Id}");

            await WaitForStatementToCompleteAsync(response.Id);

            var results = await GetStatementResultAsync(response.Id);
            var movieTitles = new List<string>();

            foreach (var row in results)
            {
                if (row.Count > 0)
                {
                    var title = row[0].StringValue;
                    Console.WriteLine($"The Movie title field is {title}");
                    movieTitles.Add(title);
                }
            }

            return movieTitles;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
```

```
            Console.WriteLine($"Couldn't query movies. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Describe a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>The statement description.</returns>
    public async Task<DescribeStatementResponse> DescribeStatementAsync(string
statementId)
    {
        try
        {
            var request = new DescribeStatementRequest
            {
                Id = statementId
            };

            var response = await
_redshiftDataClient.DescribeStatementAsync(request);
            return response;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe statement. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get the results of a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>A list of result rows.</returns>
```

```
   public async Task<List<List<Field>>> GetStatementResultAsync(string
statementId)
   {
       try
       {
           var request = new GetStatementResultRequest
           {
               Id = statementId
           };

           var response = await
_redshiftDataClient.GetStatementResultAsync(request);
           return response.Records;
       }
       catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
       {
           Console.WriteLine($"Statement not found: {ex.Message}");
           throw;
       }
       catch (Exception ex)
       {
           Console.WriteLine($"Couldn't get statement result. Here's why:
{ex.Message}");
           throw;
       }
   }

   /// <summary>
   /// Wait for a statement to complete execution.
   /// </summary>
   /// <param name="statementId">The statement ID.</param>
   /// <returns>A task representing the asynchronous operation.</returns>
   private async Task WaitForStatementToCompleteAsync(string statementId)
   {
       var status = StatusString.SUBMITTED;
       DescribeStatementResponse? response = null;

       while (status == StatusString.SUBMITTED || status == StatusString.PICKED
|| status == StatusString.STARTED)
       {
           await Task.Delay(1000); // Wait 1 second
           response = await DescribeStatementAsync(statementId);
           status = response.Status;
           Console.WriteLine($"...{status}");
```

```
        }

        if (status == StatusString.FINISHED)
        {
            Console.WriteLine("The statement is finished!");
        }
        else
        {
            var errorMessage = response?.Error ?? "Unknown error";
            Console.WriteLine($"The statement failed with status: {status}");
            Console.WriteLine($"Error message: {errorMessage}");
        }
    }

    /// <summary>
    /// Wait for a cluster to become available.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <returns>A task representing the asynchronous operation.</returns>
    public async Task WaitForClusterAvailableAsync(string clusterIdentifier)
    {
        Console.WriteLine($"Wait until {clusterIdentifier} is available. This may
 take a few minutes.");

        var startTime = DateTime.Now;
        var clusters = await DescribeClustersAsync(clusterIdentifier);

        while (clusters[0].ClusterStatus != "available")
        {
            var elapsed = DateTime.Now - startTime;
            Console.WriteLine($"Elapsed Time: {elapsed:mm\\:ss} - Waiting for
 cluster...");

            await Task.Delay(5000); // Wait 5 seconds
            clusters = await DescribeClustersAsync(clusterIdentifier);
        }

        var totalElapsed = DateTime.Now - startTime;
        Console.WriteLine($"Cluster is available! Total Elapsed Time:
 {totalElapsed:mm\\:ss}");
    }
}
```

Run an interactive scenario demonstrating Redshift basics.

```
/// <summary>
/// Amazon Redshift Getting Started Scenario.
/// </summary>
public class RedshiftBasics
{
    public static bool IsInteractive = true;
    public static RedshiftWrapper? Wrapper = null;
    public static ILogger logger = null!;
    private static readonly string _moviesFilePath = "../../../../../../
resources/sample_files/movies.json";

    /// <summary>
    /// Main method for the Amazon Redshift Getting Started scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRedshift>()
                    .AddAWSService<IAmazonRedshiftDataAPIService>()
                    .AddTransient<RedshiftWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<RedshiftBasics>();

        Wrapper = host.Services.GetRequiredService<RedshiftWrapper>();

        await RunScenarioAsync();
    }

    /// <summary>
    /// Run the complete Amazon Redshift scenario.
    /// </summary>
    public static async Task RunScenarioAsync()
    {
        // Set all variables to default values
        string userName = "awsuser";
        string userPassword = "AwsUser1000";
        string clusterIdentifier = "redshift-cluster-movies";
```

```
        var databaseName = "dev";
        int recordCount = 50;
        int year = 2013;
        try
        {
            Console.WriteLine(

 "============================================================================");
            Console.WriteLine("Welcome to the Amazon Redshift SDK Getting Started
 scenario.");
            Console.WriteLine(
                "This .NET program demonstrates how to interact with Amazon
 Redshift by using the AWS SDK for .NET.");
            Console.WriteLine("Let's get started...");
            Console.WriteLine(

 "============================================================================");

            // Step 1: Get user credentials (if interactive)
            if (IsInteractive)
            {
                Console.WriteLine("Please enter a user name for the cluster
 (default is awsuser):");
                var userInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(userInput))
                    userName = userInput;


 Console.WriteLine("============================================================================
                Console.WriteLine("Please enter a user password for the cluster
 (default is AwsUser1000):");
                var passwordInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(passwordInput))
                    userPassword = passwordInput;


 Console.WriteLine("============================================================================

                // Step 2: Get cluster identifier
                Console.WriteLine("Enter a cluster id value (default is redshift-
cluster-movies):");
                var clusterInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(clusterInput))
                    clusterIdentifier = clusterInput;
```

```
            }
            else
            {
                Console.WriteLine($"Using default values: userName={userName},
clusterIdentifier={clusterIdentifier}");
            }

            // Step 3: Create Redshift cluster
            await Wrapper!.CreateClusterAsync(clusterIdentifier, databaseName,
userName, userPassword);

Console.WriteLine("=============================================================================

            // Step 4: Wait for cluster to become available

Console.WriteLine("=============================================================================
            await Wrapper.WaitForClusterAvailableAsync(clusterIdentifier);

Console.WriteLine("=============================================================================

            // Step 5: List databases

Console.WriteLine("=============================================================================
            Console.WriteLine($" When you created {clusterIdentifier}, the dev
database is created by default and used in this scenario.");
            Console.WriteLine(" To create a custom database, you need to have a
CREATEDB privilege.");
            Console.WriteLine(" For more information, see the documentation here:
https://docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

Console.WriteLine("=============================================================================


Console.WriteLine("=============================================================================
            Console.WriteLine($"List databases in {clusterIdentifier}");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
```

```
            }
            await Wrapper.ListDatabasesAsync(clusterIdentifier, userName,
databaseName);

Console.WriteLine("================================================================================

            // Step 6: Create Movies table

Console.WriteLine("================================================================================
            Console.WriteLine("Now you will create a table named Movies.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }
            await Wrapper.CreateTableAsync(clusterIdentifier, databaseName,
userName);

Console.WriteLine("================================================================================

            // Step 7: Populate the Movies table

Console.WriteLine("================================================================================
            Console.WriteLine("Populate the Movies table using the Movies.json
file.");

            if (IsInteractive)
            {
                Console.WriteLine("Specify the number of records you would like
to add to the Movies Table.");
                Console.WriteLine("Please enter a value between 50 and 200.");
                Console.Write("Enter a value: ");

                var recordCountInput = Console.ReadLine();
                if (int.TryParse(recordCountInput, out var inputCount) &&
inputCount is >= 50 and <= 200)
                {
                    recordCount = inputCount;
                }
                else
                {
                    Console.WriteLine($"Invalid input. Using default value of
{recordCount}.");
                }
```

```
            }
            else
            {
                Console.WriteLine($"Using default record count: {recordCount}");
            }

            await PopulateMoviesTableAsync(clusterIdentifier, databaseName,
userName, recordCount);
            Console.WriteLine($"{recordCount} records were added to the Movies
table.");

Console.WriteLine("================================================================================

            // Step 8 & 9: Query movies by year

Console.WriteLine("================================================================================
            Console.WriteLine("Query the Movies table by year. Enter a value
between 2012-2014.");

            if (IsInteractive)
            {
                Console.Write("Enter a year: ");
                var yearInput = Console.ReadLine();
                if (int.TryParse(yearInput, out var inputYear) && inputYear is >=
2012 and <= 2014)
                {
                    year = inputYear;
                }
                else
                {
                    Console.WriteLine($"Invalid input. Using default value of
{year}.");
                }
            }
            else
            {
                Console.WriteLine($"Using default year: {year}");
            }

            await Wrapper.QueryMoviesByYearAsync(clusterIdentifier, databaseName,
userName, year);

Console.WriteLine("================================================================================
```

```
            // Step 10: Modify the cluster

 Console.WriteLine("=============================================================================
            Console.WriteLine("Now you will modify the Redshift cluster.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }
            await Wrapper.ModifyClusterAsync(clusterIdentifier, "wed:07:30-
wed:08:00");

 Console.WriteLine("=============================================================================

            // Step 11 & 12: Delete cluster confirmation

 Console.WriteLine("=============================================================================
            if (IsInteractive)
            {
                Console.WriteLine("Would you like to delete the Amazon Redshift
 cluster? (y/n)");
                var deleteResponse = Console.ReadLine();
                if (deleteResponse?.ToLower() == "y")
                {
                    await
 Wrapper.DeleteClusterWithoutSnapshotAsync(clusterIdentifier);
                }
            }
            else
            {
                Console.WriteLine("Deleting the Amazon Redshift cluster...");
                await
 Wrapper.DeleteClusterWithoutSnapshotAsync(clusterIdentifier);
            }

 Console.WriteLine("=============================================================================


 Console.WriteLine("=============================================================================
            Console.WriteLine("This concludes the Amazon Redshift SDK Getting
 Started scenario.");

 Console.WriteLine("=============================================================================
        }
```

```
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred during the scenario:
{ex.Message}");
            Console.WriteLine("Deleting the Amazon Redshift cluster...");
            await Wrapper!.DeleteClusterWithoutSnapshotAsync(clusterIdentifier);
            throw;
        }
    }

    /// <summary>
    /// Populate the Movies table with data from the JSON file.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="recordCount">Number of records to insert.</param>
    private static async Task PopulateMoviesTableAsync(string clusterIdentifier,
string database, string dbUser, int recordCount)
    {
        if (!File.Exists(_moviesFilePath))
        {
            throw new FileNotFoundException($"Required movies data file not found
at: {_moviesFilePath}");
        }

        var jsonContent = await File.ReadAllTextAsync(_moviesFilePath);
        var options = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true
        };
        var movies = JsonSerializer.Deserialize<List<Movie>>(jsonContent,
options);

        if (movies == null || movies.Count == 0)
        {
            throw new InvalidOperationException("Failed to parse movies JSON file
or file is empty.");
        }

        var insertCount = Math.Min(recordCount, movies.Count);

        for (int i = 0; i < insertCount; i++)
        {
```

```
            var movie = movies[i];
            await Wrapper!.InsertMovieAsync(clusterIdentifier, database, dbUser,
 i, movie.Title, movie.Year);
        }
    }

    /// <summary>
    /// Movie data model.
    /// </summary>
    private class Movie
    {
        public string Title { get; set; } = string.Empty;
        public int Year { get; set; }
    }
}
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Go

SDK for Go V2

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
package scenarios
```

```
import (
 "context"
 "encoding/json"
 "errors"
 "fmt"
 "log"
 "math/rand"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 redshift_types "github.com/aws/aws-sdk-go-v2/service/redshift/types"
 redshiftdata_types "github.com/aws/aws-sdk-go-v2/service/redshiftdata/types"
 "github.com/aws/aws-sdk-go-v2/service/secretsmanager"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/redshift/actions"

 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshiftdata"
)

// IScenarioHelper abstracts input and wait functions from a scenario so that
 they
// can be mocked for unit testing.
type IScenarioHelper interface {
 GetName() string
}

const rMax = 100000

type ScenarioHelper struct {
 Prefix string
 Random *rand.Rand
}

// GetName returns a unique name formed of a prefix and a random number.
func (helper ScenarioHelper) GetName() string {
 return fmt.Sprintf("%v%v", helper.Prefix, helper.Random.Intn(rMax))
}

// RedshiftBasicsScenario separates the steps of this scenario into individual
 functions so that
// they are simpler to read and understand.
```

```
type RedshiftBasicsScenario struct {
 sdkConfig         aws.Config
 helper            IScenarioHelper
 questioner        demotools.IQuestioner
 pauser            demotools.IPausable
 filesystem        demotools.IFileSystem
 redshiftActor     *actions.RedshiftActions
 redshiftDataActor *actions.RedshiftDataActions
 secretsmanager    *SecretsManager
}

// SecretsManager is used to retrieve username and password information from a
 secure service.
type SecretsManager struct {
 SecretsManagerClient *secretsmanager.Client
}

// RedshiftBasics constructs a new Redshift Basics runner.
func RedshiftBasics(sdkConfig aws.Config, questioner demotools.IQuestioner,
 pauser demotools.IPausable, filesystem demotools.IFileSystem, helper
 IScenarioHelper) RedshiftBasicsScenario {
 scenario := RedshiftBasicsScenario{
  sdkConfig:         sdkConfig,
  helper:            helper,
  questioner:        questioner,
  pauser:            pauser,
  filesystem:        filesystem,
  secretsmanager:    &SecretsManager{SecretsManagerClient:
 secretsmanager.NewFromConfig(sdkConfig)},
  redshiftActor:     &actions.RedshiftActions{RedshiftClient:
 redshift.NewFromConfig(sdkConfig)},
  redshiftDataActor: &actions.RedshiftDataActions{RedshiftDataClient:
 redshiftdata.NewFromConfig(sdkConfig)},
 }
 return scenario
}


// Movie makes it easier to use Movie objects given in json format.
type Movie struct {
 ID    int    `json:"id"`
 Title string `json:"title"`
 Year  int    `json:"year"`
}
```

```
// User makes it easier to get the User data back from SecretsManager and use it
 later.
type User struct {
 Username string `json:"userName"`
 Password string `json:"userPassword"`
}

// Run runs the RedshiftBasics interactive example that shows you how to use
 Amazon
// Redshift and how to interact with its common endpoints.
//
// 0. Retrieve username and password information to access Redshift.
// 1. Create a cluster.
// 2. Wait for the cluster to become available.
// 3. List the available databases in the region.
// 4. Create a table named "Movies" in the "dev" database.
// 5. Populate the movies table from the "movies.json" file.
// 6. Query the movies table by year.
// 7. Modify the cluster's maintenance window.
// 8. Optionally clean up all resources created during this demo.
//
// This example creates an Amazon Redshift service client from the specified
 sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func (runner *RedshiftBasicsScenario) Run(ctx context.Context) {

 user := User{}
 secretId := "s3express/basics/secrets"
 clusterId := "demo-cluster-1"
 maintenanceWindow := "wed:07:30-wed:08:00"
 databaseName := "dev"
 tableName := "Movies"
 fileName := "Movies.json"
 nodeType := "ra3.xlplus"
 clusterType := "single-node"

 defer func() {
  if r := recover(); r != nil {
```

```
    log.Println("Something went wrong with the demo.")
    _, isMock := runner.questioner.(*demotools.MockQuestioner)
    if isMock || runner.questioner.AskBool("Do you want to see the full error
 message (y/n)?", "y") {
     log.Println(r)
    }
    runner.cleanUpResources(ctx, clusterId, databaseName, tableName,
 user.Username, runner.questioner)
   }
 }()

 // Retrieve the userName and userPassword from SecretsManager
 output, err := runner.secretsmanager.SecretsManagerClient.GetSecretValue(ctx,
 &secretsmanager.GetSecretValueInput{
  SecretId: aws.String(secretId),
 })
 if err != nil {
  log.Printf("There was a problem getting the secret value: %s", err)
  log.Printf("Please make sure to create a secret named 's3express/basics/
secrets' with keys of 'userName' and 'userPassword'.")
  panic(err)
 }

 err = json.Unmarshal([]byte(*output.SecretString), &user)
 if err != nil {
  log.Printf("There was a problem parsing the secret value from JSON: %s", err)
  panic(err)
 }

 // Create the Redshift cluster
 _, err = runner.redshiftActor.CreateCluster(ctx, clusterId, user.Username,
 user.Password, nodeType, clusterType, true)
 if err != nil {
  var clusterAlreadyExistsFault *redshift_types.ClusterAlreadyExistsFault
  if errors.As(err, &clusterAlreadyExistsFault) {
   log.Println("Cluster already exists. Continuing.")
  } else {
   log.Println("Error creating cluster.")
   panic(err)
  }
 }

 // Wait for the cluster to become available
```

```
waiter :=
redshift.NewClusterAvailableWaiter(runner.redshiftActor.RedshiftClient)
err = waiter.Wait(ctx, &redshift.DescribeClustersInput{
 ClusterIdentifier: aws.String(clusterId),
}, 5*time.Minute)
if err != nil {
 log.Println("An error occurred waiting for the cluster.")
 panic(err)
}

// Get some info about the cluster
describeOutput, err := runner.redshiftActor.DescribeClusters(ctx, clusterId)
if err != nil {
 log.Println("Something went wrong trying to get information about the
cluster.")
 panic(err)
}
log.Println("Here's some information about the cluster.")
log.Printf("The cluster's status is %s",
*describeOutput.Clusters[0].ClusterStatus)
log.Printf("The cluster was created at %s",
*describeOutput.Clusters[0].ClusterCreateTime)

// List databases
log.Println("List databases in", clusterId)
runner.questioner.Ask("Press Enter to continue...")
err = runner.redshiftDataActor.ListDatabases(ctx, clusterId, databaseName,
user.Username)
if err != nil {
 log.Printf("Failed to list databases: %v\n", err)
 panic(err)
}

// Create the "Movies" table
log.Println("Now you will create a table named " + tableName + ".")
runner.questioner.Ask("Press Enter to continue...")
err = nil
result, err := runner.redshiftDataActor.CreateTable(ctx, clusterId,
databaseName, tableName, user.Username, runner.pauser, []string{"title
VARCHAR(256)", "year INT"})
if err != nil {
 log.Printf("Failed to create table: %v\n", err)
 panic(err)
}
```

```
 describeInput := redshiftdata.DescribeStatementInput{
  Id: result.Id,
 }
 query := actions.RedshiftQuery{
  Context: ctx,
  Input:   describeInput,
  Result:  result,
 }
 err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
 if err != nil {
  log.Printf("Failed to execute query: %v\n", err)
  panic(err)
 }
 log.Printf("Successfully executed query\n")

 // Populate the "Movies" table
 runner.PopulateMoviesTable(ctx, clusterId, databaseName, tableName,
 user.Username, fileName)

 // Query the "Movies" table by year
 log.Println("Query the Movies table by year.")
 year := runner.questioner.AskInt(
  fmt.Sprintf("Enter a value between %v and %v:", 2012, 2014),
  demotools.InIntRange{Lower: 2012, Upper: 2014})
 runner.QueryMoviesByYear(ctx, clusterId, databaseName, tableName, user.Username,
 year)

 // Modify the cluster's maintenance window
 runner.redshiftActor.ModifyCluster(ctx, clusterId, maintenanceWindow)

 // Delete the Redshift cluster if confirmed
 runner.cleanUpResources(ctx, clusterId, databaseName, tableName, user.Username,
 runner.questioner)

 log.Println("Thanks for watching!")
}

// cleanUpResources asks the user if they would like to delete each resource
 created during the scenario, from most
// impactful to least impactful. If any choice to delete is made, further
 deletion attempts are skipped.
```

```
func (runner *RedshiftBasicsScenario) cleanUpResources(ctx context.Context,
 clusterId string, databaseName string, tableName string, userName string,
 questioner demotools.IQuestioner) {
 deleted := false
 var err error = nil
 if questioner.AskBool("Do you want to delete the entire cluster? This will clean
 up all resources. (y/n)", "y") {
  deleted, err = runner.redshiftActor.DeleteCluster(ctx, clusterId)
  if err != nil {
   log.Printf("Error deleting cluster: %v", err)
  }
 }
 if !deleted && questioner.AskBool("Do you want to delete the dev table? This
 will clean up all inserted records but keep your cluster intact. (y/n)", "y") {
  deleted, err = runner.redshiftDataActor.DeleteTable(ctx, clusterId,
 databaseName, tableName, userName)
  if err != nil {
   log.Printf("Error deleting movies table: %v", err)
  }
 }
 if !deleted && questioner.AskBool("Do you want to delete all rows in the Movies
 table? This will clean up all inserted records but keep your cluster and table
 intact. (y/n)", "y") {
  deleted, err = runner.redshiftDataActor.DeleteDataRows(ctx, clusterId,
 databaseName, tableName, userName, runner.pauser)
  if err != nil {
   log.Printf("Error deleting data rows: %v", err)
  }
 }
 if !deleted {
  log.Print("Please manually delete any unwanted resources.")
 }
}


// loadMoviesFromJSON takes the <fileName> file and populates a slice of Movie
 objects.
func (runner *RedshiftBasicsScenario) loadMoviesFromJSON(fileName string,
 filesystem demotools.IFileSystem) ([]Movie, error) {
 file, err := filesystem.OpenFile("../../resources/sample_files/" + fileName)
 if err != nil {
  return nil, err
 }
 defer filesystem.CloseFile(file)
```

```
 var movies []Movie
 err = json.NewDecoder(file).Decode(&movies)
 if err != nil {
  return nil, err
 }

 return movies, nil
}



// PopulateMoviesTable reads data from the <fileName> file and inserts records
 into the "Movies" table.
func (runner *RedshiftBasicsScenario) PopulateMoviesTable(ctx context.Context,
 clusterId string, databaseName string, tableName string, userName string,
 fileName string) {
 log.Println("Populate the " + tableName + " table using the " + fileName + "
 file.")
 numRecords := runner.questioner.AskInt(
  fmt.Sprintf("Enter a value between %v and %v:", 10, 100),
  demotools.InIntRange{Lower: 10, Upper: 100})

 movies, err := runner.loadMoviesFromJSON(fileName, runner.filesystem)
 if err != nil {
  log.Printf("Failed to load movies from JSON: %v\n", err)
  panic(err)
 }

 var sqlStatements []string

 for i, movie := range movies {
  if i >= numRecords {
   break
  }

  sqlStatement := fmt.Sprintf(`INSERT INTO %s (title, year) VALUES ('%s', %d);`,
   tableName,
   strings.Replace(movie.Title, "'", "''", -1), // Double any single quotes to
 escape them
   movie.Year)

  sqlStatements = append(sqlStatements, sqlStatement)
 }
```

```
 input := &redshiftdata.BatchExecuteStatementInput{
  ClusterIdentifier: aws.String(clusterId),
  Database:          aws.String(databaseName),
  DbUser:            aws.String(userName),
  Sqls:              sqlStatements,
 }

 result, err := runner.redshiftDataActor.ExecuteBatchStatement(ctx, *input)
 if err != nil {
  log.Printf("Failed to execute batch statement: %v\n", err)
  panic(err)
 }

 describeInput := redshiftdata.DescribeStatementInput{
  Id: result.Id,
 }

 query := actions.RedshiftQuery{
  Context: ctx,
  Result:  result,
  Input:   describeInput,
 }
 err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
 if err != nil {
  log.Printf("Failed to execute batch insert query: %v\n", err)
  return
 }
 log.Printf("Successfully executed batch statement\n")

 log.Printf("%d records were added to the Movies table.\n", numRecords)
}



// QueryMoviesByYear retrieves only movies from the "Movies" table which match
 the given year.
func (runner *RedshiftBasicsScenario) QueryMoviesByYear(ctx context.Context,
 clusterId string, databaseName string, tableName string, userName string, year
 int) {

 sqlStatement := fmt.Sprintf(`SELECT title FROM %s WHERE year = %d;`, tableName,
 year)
```

```
input := &redshiftdata.ExecuteStatementInput{
 ClusterIdentifier: aws.String(clusterId),
 Database:          aws.String(databaseName),
 DbUser:            aws.String(userName),
 Sql:               aws.String(sqlStatement),
}

result, err := runner.redshiftDataActor.ExecuteStatement(ctx, *input)
if err != nil {
 log.Printf("Failed to query movies: %v\n", err)
 panic(err)
}

log.Println("The identifier of the statement is ", *result.Id)

describeInput := redshiftdata.DescribeStatementInput{
 Id: result.Id,
}

query := actions.RedshiftQuery{
 Context: ctx,
 Input:   describeInput,
 Result:  result,
}
err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
if err != nil {
 log.Printf("Failed to execute query: %v\n", err)
 panic(err)
}
log.Printf("Successfully executed query\n")

getResultOutput, err := runner.redshiftDataActor.GetStatementResult(ctx,
*result.Id)
if err != nil {
 log.Printf("Failed to query movies: %v\n", err)
 panic(err)
}
for _, row := range getResultOutput.Records {
 for _, col := range row {
  title, ok := col.(*redshiftdata_types.FieldMemberStringValue)
  if !ok {
   log.Println("Failed to parse the field")
  } else {
   log.Printf("The Movie title field is %s\n", title.Value)
```

```
    }
   }
  }
 }
```

- For API details, see the following topics in *AWS SDK for Go API Reference*.
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Run an interactive scenario demonstrating Amazon Redshift features.

```
import com.example.redshift.User;
import com.google.gson.Gson;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.redshift.model.ClusterAlreadyExistsException;
import software.amazon.awssdk.services.redshift.model.CreateClusterResponse;
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;
import software.amazon.awssdk.services.redshift.model.ModifyClusterResponse;
```

```
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import
 software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.RedshiftDataException;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 *
 *  This example requires an AWS Secrets Manager secret that contains the
 *  database credentials. If you do not create a
 *  secret that specifies user name and password, this example will not work. For
 details, see:
 *
 *  https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 This Java example performs these tasks:
 *
 * 1. Prompts the user for a unique cluster ID or use the default value.
 * 2. Creates a Redshift cluster with the specified or default cluster Id value.
 * 3. Waits until the Redshift cluster is available for use.
 * 4. Lists all databases using a pagination API call.
 * 5. Creates a table named "Movies" with fields ID, title, and year.
 * 6. Inserts a specified number of records into the "Movies" table by reading
 the Movies JSON file.
 * 7. Prompts the user for a movie release year.
 * 8. Runs a SQL query to retrieve movies released in the specified year.
 * 9. Modifies the Redshift cluster.
 * 10. Prompts the user for confirmation to delete the Redshift cluster.
 * 11. If confirmed, deletes the specified Redshift cluster.
```

```
 */

public class RedshiftScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftScenario.class);

    static RedshiftActions redshiftActions = new RedshiftActions();
    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:
                <jsonFilePath> <secretName>\s

            Where:
                jsonFilePath - The path to the Movies JSON file (you can locate
 that file in ../../../resources/sample_files/movies.json)
                secretName - The name of the secret that belongs to Secret
 Manager that stores the user name and password used in this scenario.
            """;

        if (args.length != 2) {
            logger.info(usage);
            return;
        }

        String jsonFilePath = args[0];
        String secretName = args[1];
        Scanner scanner = new Scanner(System.in);
        logger.info(DASHES);
        logger.info("Welcome to the Amazon Redshift SDK Basics scenario.");
        logger.info("""
            This Java program demonstrates how to interact with Amazon Redshift
 by using the AWS SDK for Java (v2).\s
            Amazon Redshift is a fully managed, petabyte-scale data warehouse
 service hosted in the cloud.

            The program's primary functionalities include cluster creation,
 verification of cluster readiness,\s
            list databases, table creation, data population within the table, and
 execution of SQL statements.
            Furthermore, it demonstrates the process of querying data from the
 Movie table.\s
```

```
            Upon completion of the program, all AWS resources are cleaned up.
            """);

        logger.info("Lets get started...");
        logger.info("""
            First, we will retrieve the user name and password from Secrets
 Manager.

            Using Amazon Secrets Manager to store Redshift credentials provides
 several security benefits.
            It allows you to securely store and manage sensitive information,
 such as passwords, API keys, and
            database credentials, without embedding them directly in your
 application code.

            More information can be found here:

            https://docs.aws.amazon.com/secretsmanager/latest/userguide/
integrating_how-services-use-secrets_RS.html
            """);
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            runScenario(user, scanner, jsonFilePath);
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (Throwable e) {
            throw new RuntimeException(e);
        }
    }

    private static void runScenario(User user, Scanner scanner,  String
 jsonFilePath) throws Throwable {
        String databaseName = "dev";
        System.out.println(DASHES);
        logger.info("Create a Redshift Cluster");
        logger.info("A Redshift cluster refers to the collection of computing
 resources and storage that work together to process and analyze large volumes of
 data.");
```

```
        logger.info("Enter a cluster id value or accept the default by hitting
Enter (default is redshift-cluster-movies): ");
        String userClusterId = scanner.nextLine();
        String clusterId = userClusterId.isEmpty() ? "redshift-cluster-movies" :
userClusterId;
        try {
            CompletableFuture<CreateClusterResponse> future =
redshiftActions.createClusterAsync(clusterId, user.getUserName(),
user.getUserPassword());
            CreateClusterResponse response = future.join();
            logger.info("Cluster successfully created. Cluster Identifier {} ",
response.cluster().clusterIdentifier());

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof ClusterAlreadyExistsException) {
                logger.info("The Cluster {} already exists. Moving on...",
clusterId);
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Wait until {} is available.", clusterId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
redshiftActions.waitForClusterReadyAsync(clusterId);
            future.join();
            logger.info("Cluster is ready!");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftException redshiftEx) {
                logger.info("Redshift error occurred: Error message: {}, Error
code {}", redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        String databaseInfo = """
            When you created $clusteridD, the dev database is created by default
 and used in this scenario.\s

            To create a custom database, you need to have a CREATEDB privilege.\s
            For more information, see the documentation here: https://
docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.
          """.replace("$clusteridD", clusterId);

        logger.info(databaseInfo);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("List databases in {} ",clusterId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
 redshiftActions.listAllDatabasesAsync(clusterId, user.getUserName(), "dev");
            future.join();
            logger.info("Databases listed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.error("Redshift Data error occurred: {} Error code: {}",
 redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.error("An unexpected error occurred: {}",
 rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Now you will create a table named Movies.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ExecuteStatementResponse> future =
 redshiftActions.createTableAsync(clusterId, databaseName, user.getUserName());
            future.join();
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Populate the Movies table using the Movies.json file.");
        logger.info("Specify the number of records you would like to add to the
Movies Table.");
        logger.info("Please enter a value between 50 and 200.");
        int numRecords;
        do {
            logger.info("Enter a value: ");
            while (!scanner.hasNextInt()) {
                logger.info("Invalid input. Please enter a value between 50 and
200.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            numRecords = scanner.nextInt();
        } while (numRecords < 50 || numRecords > 200);
        try {
            redshiftActions.popTableAsync(clusterId, databaseName,
user.getUserName(), jsonFilePath, numRecords).join();  // Wait for the operation
to complete
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Query the Movies table by year. Enter a value between
2012-2014.");
        int movieYear;
        do {
            logger.info("Enter a year: ");
            while (!scanner.hasNextInt()) {
                logger.info("Invalid input. Please enter a valid year between
2012 and 2014.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            movieYear = scanner.nextInt();
            scanner.nextLine();
        } while (movieYear < 2012 || movieYear > 2014);

        String id;
        try {
            CompletableFuture<String> future =
redshiftActions.queryMoviesByYearAsync(databaseName, user.getUserName(),
movieYear, clusterId);
            id = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        logger.info("The identifier of the statement is " + id);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
redshiftActions.checkStatementAsync(id);
            future.join();

        } catch (RuntimeException rt) {
```

```
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = redshiftActions.getResultsAsync(id);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Now you will modify the Redshift cluster.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ModifyClusterResponse> future =
redshiftActions.modifyClusterAsync(clusterId);;
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
```

```
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Would you like to delete the Amazon Redshift cluster? (y/
n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete {} ", clusterId);
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<DeleteClusterResponse> future =
 redshiftActions.deleteRedshiftClusterAsync(clusterId);;
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof RedshiftDataException redshiftEx) {
                    logger.info("Redshift Data error occurred: {} Error code:
 {}", redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: {}",
 rt.getMessage());
                }
                throw cause;
            }
        } else {
            logger.info("The {}  was not deleted", clusterId);
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("This concludes the Amazon Redshift SDK Basics scenario.");
        logger.info(DASHES);
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_EAST_1;
        return SecretsManagerClient.builder()
            .region(region)
            .build();
    }
```

```
    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }

    // Get the Amazon Redshift credentials from AWS Secrets Manager.
    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
 secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }
}
```

A wrapper class for Amazon Redshift SDK methods.

```
public class RedshiftActions {

    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftActions.class);
    private static RedshiftDataAsyncClient redshiftDataAsyncClient;

    private static RedshiftAsyncClient redshiftAsyncClient;

    private static RedshiftAsyncClient getAsyncClient() {
        if (redshiftAsyncClient == null) {
```

```
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            redshiftAsyncClient = RedshiftAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return redshiftAsyncClient;
    }

    private static RedshiftDataAsyncClient getAsyncDataClient() {
        if (redshiftDataAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            redshiftDataAsyncClient = RedshiftDataAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return redshiftDataAsyncClient;
```

```
    }

    /**
     * Creates a new Amazon Redshift cluster asynchronously.
     * @param clusterId     the unique identifier for the cluster
     * @param username      the username for the administrative user
     * @param userPassword  the password for the administrative user
     * @return a CompletableFuture that represents the asynchronous operation of
creating the cluster
     * @throws RuntimeException if the cluster creation fails
     */
    public CompletableFuture<CreateClusterResponse> createClusterAsync(String
clusterId, String username, String userPassword) {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(username)
            .masterUserPassword(userPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        return getAsyncClient().createCluster(clusterRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Created cluster ");
                } else {
                    throw new RuntimeException("Failed to create cluster: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Waits asynchronously for the specified cluster to become available.
     * @param clusterId the identifier of the cluster to wait for
     * @return a {@link CompletableFuture} that completes when the cluster is
ready
     */
    public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
        DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();
```

```
        logger.info("Waiting for cluster to become available. This may take a few
minutes.");
        long startTime = System.currentTimeMillis();

        // Recursive method to poll the cluster status.
        return checkClusterStatusAsync(clustersRequest, startTime);
    }

    private CompletableFuture<Void>
checkClusterStatusAsync(DescribeClustersRequest clustersRequest, long startTime)
{
        return getAsyncClient().describeClusters(clustersRequest)
            .thenCompose(clusterResponse -> {
                List<Cluster> clusterList = clusterResponse.clusters();
                boolean clusterReady = false;
                for (Cluster cluster : clusterList) {
                    if ("available".equals(cluster.clusterStatus())) {
                        clusterReady = true;
                        break;
                    }
                }

                if (clusterReady) {
                    logger.info(String.format("Cluster is available!"));
                    return CompletableFuture.completedFuture(null);
                } else {
                    long elapsedTimeMillis = System.currentTimeMillis() -
startTime;
                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;
                    System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
                    System.out.flush();

                    // Wait 1 second before the next status check
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
```

```
                    }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
                }
            }).exceptionally(exception -> {
                throw new RuntimeException("Failed to get cluster status: " +
exception.getMessage(), exception);
            });
    }

    /**
     * Lists all databases asynchronously for the specified cluster, database
user, and database.
     * @param clusterId the identifier of the cluster to list databases for
     * @param dbUser the database user to use for the list databases request
     * @param database the database to list databases for
     * @return a {@link CompletableFuture} that completes when the database
listing is complete, or throws a {@link RuntimeException} if there was an error
     */
    public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();

        // Asynchronous paginator for listing databases.
        ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
        CompletableFuture<Void> future = databasesPaginator.subscribe(response ->
{
            response.databases().forEach(db -> {
                logger.info("The database name is {} ", db);
            });
        });

        // Return the future for asynchronous handling.
        return future.exceptionally(exception -> {
            throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
        });
    }

    /**
```

```
    * Creates an asynchronous task to execute a SQL statement for creating a new
table.
    *
    * @param clusterId    the identifier of the Amazon Redshift cluster
    * @param databaseName the name of the database to create the table in
    * @param userName     the username to use for the database connection
    * @return a {@link CompletableFuture} that completes with the result of the
SQL statement execution
    * @throws RuntimeException if there is an error creating the table
    */
   public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
       ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
           .clusterIdentifier(clusterId)
           .dbUser(userName)
           .database(databaseName)
           .sql("CREATE TABLE Movies (" +
               "id INT PRIMARY KEY, " +
               "title VARCHAR(100), " +
               "year INT)")
           .build();

       return getAsyncDataClient().executeStatement(createTableRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
               } else {
                   logger.info("Table created: Movies");
               }
           });
   }

   /**
    * Asynchronously pops a table from a JSON file.
    *
    * @param clusterId   the ID of the cluster
    * @param databaseName the name of the database
    * @param userName    the username
    * @param fileName    the name of the JSON file
    * @param number      the number of records to process
    * @return a CompletableFuture that completes with the number of records
added to the Movies table
```

```
    */
   public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
       return CompletableFuture.supplyAsync(() -> {
               try {
                   JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                   JsonNode rootNode = new ObjectMapper().readTree(parser);
                   Iterator<JsonNode> iter = rootNode.iterator();
                   return iter;
               } catch (IOException e) {
                   throw new RuntimeException("Failed to read or parse JSON
file: " + e.getMessage(), e);
               }
           }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
           .whenComplete((result, exception) -> {
               if (exception != null) {
                   logger.info("Error {} ", exception.getMessage());
               } else {
                   logger.info("{} records were added to the Movies table." ,
result);
               }
           });
   }

   private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
       return CompletableFuture.supplyAsync(() -> {
           int t = 0;
           try {
               while (iter.hasNext()) {
                   if (t == number)
                       break;
                   JsonNode currentNode = iter.next();
                   int year = currentNode.get("year").asInt();
                   String title = currentNode.get("title").asText();

                   // Use SqlParameter to avoid SQL injection.
                   List<SqlParameter> parameterList = new ArrayList<>();
                   String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";
                   SqlParameter idParam = SqlParameter.builder()
                       .name("id")
```

```
                        .value(String.valueOf(t))
                        .build();

                    SqlParameter titleParam = SqlParameter.builder()
                        .name("title")
                        .value(title)
                        .build();

                    SqlParameter yearParam = SqlParameter.builder()
                        .name("year")
                        .value(String.valueOf(year))
                        .build();
                    parameterList.add(idParam);
                    parameterList.add(titleParam);
                    parameterList.add(yearParam);

                    ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                        .clusterIdentifier(clusterId)
                        .sql(sqlStatement)
                        .database(databaseName)
                        .dbUser(userName)
                        .parameters(parameterList)
                        .build();


getAsyncDataClient().executeStatement(insertStatementRequest);
                    logger.info("Inserted: " + title + " (" + year + ")");
                    t++;
                }
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
            }
            return t;
        });
    }

    /**
     * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
     *
     * @param sqlId the ID of the SQL statement to check
```

```
     * @return a {@link CompletableFuture} that completes when the SQL
statement's status is either "FINISHED" or "FAILED"
     */
    public CompletableFuture<Void> checkStatementAsync(String sqlId) {
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();

        return getAsyncDataClient().describeStatement(statementRequest)
            .thenCompose(response -> {
                String status = response.statusAsString();
                logger.info("... Status: {} ", status);

                if ("FAILED".equals(status)) {
                    throw new RuntimeException("The Query Failed. Ending
program");
                } else if ("FINISHED".equals(status)) {
                    return CompletableFuture.completedFuture(null);
                } else {
                    // Sleep for 1 second and recheck status
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
Recursively call until status is FINISHED or FAILED
                }
            }).whenComplete((result, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    logger.info("Error: {} ", exception.getMessage());
                } else {
                    logger.info("The statement is finished!");
                }
            });
    }

    /**
     * Asynchronously retrieves the results of a statement execution.
     *
```

```
    * @param statementId the ID of the statement for which to retrieve the
results
    * @return a {@link CompletableFuture} that completes when the statement
result has been processed
    */
   public CompletableFuture<Void> getResultsAsync(String statementId) {
       GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
           .id(statementId)
           .build();

       return getAsyncDataClient().getStatementResult(resultRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.info("Error getting statement result {} ",
exception.getMessage());
                   throw new RuntimeException("Error getting statement result: "
+ exception.getMessage(), exception);
               }

               // Extract and print the field values using streams if the
response is valid.
               response.records().stream()
                   .flatMap(List::stream)
                   .map(Field::stringValue)
                   .filter(value -> value != null)
                   .forEach(value -> System.out.println("The Movie title field
is " + value));

               return response;
           }).thenAccept(response -> {
               // Optionally add more logic here if needed after handling the
response
           });
   }


   /**
    * Asynchronously queries movies by a given year from a Redshift database.
    *
    * @param database    the name of the database to query
    * @param dbUser      the user to connect to the database with
    * @param year        the year to filter the movies by
    * @param clusterId   the identifier of the Redshift cluster to connect to
```

```
    * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
    */
   public CompletableFuture<String> queryMoviesByYearAsync(String database,
                                                           String dbUser,
                                                           int year,
                                                           String
clusterId) {

       String sqlStatement = "SELECT * FROM Movies WHERE year = :year";
       SqlParameter yearParam = SqlParameter.builder()
           .name("year")
           .value(String.valueOf(year))
           .build();

       ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
           .clusterIdentifier(clusterId)
           .database(database)
           .dbUser(dbUser)
           .parameters(yearParam)
           .sql(sqlStatement)
           .build();

       return CompletableFuture.supplyAsync(() -> {
           try {
               ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
               return response.id();
           } catch (RedshiftDataException e) {
               throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
           }
       }).exceptionally(exception -> {
           logger.info("Error: {}", exception.getMessage());
           return "";
       });
   }

   /**
    * Modifies an Amazon Redshift cluster asynchronously.
    *
    * @param clusterId the identifier of the cluster to be modified
```

```
    * @return a {@link CompletableFuture} that completes when the cluster
modification is complete
    */
   public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
clusterId) {
       ModifyClusterRequest modifyClusterRequest =
ModifyClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .preferredMaintenanceWindow("wed:07:30-wed:08:00")
           .build();

       return getAsyncClient().modifyCluster(modifyClusterRequest)
           .whenComplete((clusterResponse, exception) -> {
               if (exception != null) {
                   if (exception.getCause() instanceof RedshiftException) {
                       logger.info("Error: {} ", exception.getMessage());
                   } else {
                       logger.info("Unexpected error: {} ",
exception.getMessage());
                   }
               } else {
                   logger.info("The modified cluster was successfully modified
and has "
                       + clusterResponse.cluster().preferredMaintenanceWindow()
+ " as the maintenance window");
               }
           });
   }

   /**
    * Deletes a Redshift cluster asynchronously.
    *
    * @param clusterId the identifier of the Redshift cluster to be deleted
    * @return a {@link CompletableFuture} that represents the asynchronous
operation of deleting the Redshift cluster
    */
   public CompletableFuture<DeleteClusterResponse>
deleteRedshiftClusterAsync(String clusterId) {
       DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .skipFinalClusterSnapshot(true)
           .build();
```

```
        return getAsyncClient().deleteCluster(deleteClusterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {}", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {}",
 exception.getMessage());
                    }
                } else {
                    // Handle successful response
                    logger.info("The status is {}",
 response.cluster().clusterStatus());
                }
            });
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class RedshiftScenario:
    """Runs an interactive scenario that shows how to get started with
 Redshift."""

    def __init__(self, redshift_wrapper, redshift_data_wrapper):
        self.redshift_wrapper = redshift_wrapper
        self.redshift_data_wrapper = redshift_data_wrapper

    def redhift_scenario(self, json_file_path):
        database_name = "dev"

        print(DASHES)
        print("Welcome to the Amazon Redshift SDK Getting Started example.")
        print(
            """
      This Python program demonstrates how to interact with Amazon Redshift
      using the AWS SDK for Python (Boto3).

      Amazon Redshift is a fully managed, petabyte-scale data warehouse
      service hosted in the cloud.

      The program's primary functionalities include cluster creation,
      verification of cluster readiness, listing databases, table creation,
      populating data within the table, and executing SQL statements.

      It also demonstrates querying data from the Movies table.

      Upon completion, all AWS resources are cleaned up.
    """
        )
        if not os.path.isfile(json_file_path):
            logging.error(f"The file {json_file_path} does not exist.")
            return

        print("Let's get started...")
        user_name = q.ask("Please enter your user name (default is awsuser):")
        user_name = user_name if user_name else "awsuser"

        print(DASHES)
        user_password = q.ask(
            "Please enter your user password (default is AwsUser1000):"
        )
        user_password = user_password if user_password else "AwsUser1000"
```

```
        print(DASHES)
        print(
            """A Redshift cluster refers to the collection of computing resources
 and storage that work
            together to process and analyze large volumes of data."""
        )
        cluster_id = q.ask(
            "Enter a cluster identifier value (default is redshift-cluster-
movies): "
        )
        cluster_id = cluster_id if cluster_id else "redshift-cluster-movies"

        self.redshift_wrapper.create_cluster(
            cluster_id, "ra3.4xlarge", user_name, user_password, True, 2
        )

        print(DASHES)
        print(f"Wait until {cluster_id} is available. This may take a few
 minutes...")
        q.ask("Press Enter to continue...")

        self.wait_cluster_available(cluster_id)

        print(DASHES)

        print(
            f"""
       When you created {cluster_id}, the dev database is created by default and
 used in this scenario.

       To create a custom database, you need to have a CREATEDB privilege.
       For more information, see the documentation here:
       https://docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.
      """
        )
        q.ask("Press Enter to continue...")
        print(DASHES)

        print(DASHES)
        print(f"List databases in {cluster_id}")
        q.ask("Press Enter to continue...")
        databases = self.redshift_data_wrapper.list_databases(
            cluster_id, database_name, user_name
```

```
        )
        print(f"The cluster contains {len(databases)} database(s).")
        for database in databases:
            print(f"    Database: {database}")
        print(DASHES)

        print(DASHES)
        print("Now you will create a table named Movies.")
        q.ask("Press Enter to continue...")

        self.create_table(cluster_id, database_name, user_name)

        print(DASHES)

        print("Populate the Movies table using the Movies.json file.")
        print(
            "Specify the number of records you would like to add to the Movies
Table."
        )
        print("Please enter a value between 50 and 200.")

        while True:
            try:
                num_records = int(q.ask("Enter a value: ", q.is_int))
                if 50 <= num_records <= 200:
                    break
                else:
                    print("Invalid input. Please enter a value between 50 and
200.")
            except ValueError:
                print("Invalid input. Please enter a value between 50 and 200.")

        self.populate_table(
            cluster_id, database_name, user_name, json_file_path, num_records
        )

        print(DASHES)
        print("Query the Movies table by year. Enter a value between 2012-2014.")

        while True:
            movie_year = int(q.ask("Enter a year: ", q.is_int))
            if 2012 <= movie_year <= 2014:
                break
            else:
```

```
                print("Invalid input. Please enter a valid year between 2012 and
2014.")

        # Function to query database
        sql_id = self.query_movies_by_year(
            database_name, user_name, movie_year, cluster_id
        )

        print(f"The identifier of the statement is {sql_id}")

        print("Checking statement status...")
        self.wait_statement_finished(sql_id)
        result = self.redshift_data_wrapper.get_statement_result(sql_id)

        self.display_movies(result)

        print(DASHES)

        print(DASHES)
        print("Now you will modify the Redshift cluster.")
        q.ask("Press Enter to continue...")

        preferred_maintenance_window = "wed:07:30-wed:08:00"
        self.redshift_wrapper.modify_cluster(cluster_id,
preferred_maintenance_window)

        print(DASHES)

        print(DASHES)
        delete = q.ask("Do you want to delete the cluster? (y/n) ", q.is_yesno)

        if delete:
            print(f"You selected to delete {cluster_id}")
            q.ask("Press Enter to continue...")
            self.redshift_wrapper.delete_cluster(cluster_id)
        else:
            print(f"Cluster {cluster_id}cluster_id was not deleted")

        print(DASHES)
        print("This concludes the Amazon Redshift SDK Getting Started scenario.")
        print(DASHES)

    def create_table(self, cluster_id, database, username):
        self.redshift_data_wrapper.execute_statement(
```

```
            cluster_identifier=cluster_id,
            database_name=database,
            user_name=username,
            sql="CREATE TABLE Movies (statement_id INT PRIMARY KEY, title
VARCHAR(100), year INT)",
        )

        print("Table created: Movies")


    def populate_table(self, cluster_id, database, username, file_name, number):
        with open(file_name) as f:
            data = json.load(f)

        i = 0
        for record in data:
            if i == number:
                break

            statement_id = i
            title = record["title"]
            year = record["year"]
            i = i + 1
            parameters = [
                {"name": "statement_id", "value": str(statement_id)},
                {"name": "title", "value": title},
                {"name": "year", "value": str(year)},
            ]

            self.redshift_data_wrapper.execute_statement(
                cluster_identifier=cluster_id,
                database_name=database,
                user_name=username,
                sql="INSERT INTO Movies VALUES(:statement_id, :title, :year)",
                parameter_list=parameters,
            )

        print(f"{i} records inserted into Movies table")

    def wait_cluster_available(self, cluster_id):
        """
        Waits for a cluster to be available.

        :param cluster_id: The cluster identifier.
```

```
        Note: The cluster_available waiter can also be used.
        It is not used in this case to allow an elapsed time message.
        """
        cluster_ready = False
        start_time = time.time()

        while not cluster_ready:
            time.sleep(30)
            cluster = self.redshift_wrapper.describe_clusters(cluster_id)
            status = cluster[0]["ClusterStatus"]
            if status == "available":
                cluster_ready = True
            elif status != "creating":
                raise Exception(
                    f"Cluster {cluster_id} creation failed with status {status}."
                )

            elapsed_seconds = int(round(time.time() - start_time))
            minutes = int(elapsed_seconds // 60)
            seconds = int(elapsed_seconds % 60)

            print(f"Elapsed Time: {minutes}:{seconds:02d} - status {status}...")

            if minutes > 30:
                raise Exception(
                    f"Cluster {cluster_id} is not available after 30 minutes."
                )

    def query_movies_by_year(self, database, username, year, cluster_id):
        sql = "SELECT * FROM Movies WHERE year = :year"

        params = [{"name": "year", "value": str(year)}]

        response = self.redshift_data_wrapper.execute_statement(
            cluster_identifier=cluster_id,
            database_name=database,
            user_name=username,
            sql=sql,
            parameter_list=params,
        )

        return response["Id"]
```

```
    @staticmethod
    def display_movies(response):
        metadata = response["ColumnMetadata"]
        records = response["Records"]

        title_column_index = None
        for i in range(len(metadata)):
            if metadata[i]["name"] == "title":
                title_column_index = i
                break

        if title_column_index is None:
            print("No title column found.")
            return

        print(f"Found {len(records)} movie(s).")
        for record in records:
            print(f"   {record[title_column_index]['stringValue']}")

    def wait_statement_finished(self, sql_id):
        while True:
            time.sleep(1)
            response = self.redshift_data_wrapper.describe_statement(sql_id)
            status = response["Status"]
            print(f"Statement status is {status}.")

            if status == "FAILED":
                print(f"The query failed because {response['Error']}. Ending
 program")
                raise Exception("The Query Failed. Ending program")
            elif status == "FINISHED":
                break
```

Main function showing scenario implementation.

```
def main():
    redshift_client = boto3.client("redshift")
    redshift_data_client = boto3.client("redshift-data")
    redshift_wrapper = RedshiftWrapper(redshift_client)
    redshift_data_wrapper = RedshiftDataWrapper(redshift_data_client)
```

```
    redshift_scenario = RedshiftScenario(redshift_wrapper, redshift_data_wrapper)
    redshift_scenario.redhift_scenario(
        f"{os.path.dirname(__file__)}/../../../resources/sample_files/
movies.json"
    )
```

The wrapper functions used in the scenario.

```
    def create_cluster(
        self,
        cluster_identifier,
        node_type,
        master_username,
        master_user_password,
        publicly_accessible,
        number_of_nodes,
    ):
        """
        Creates a cluster.

        :param cluster_identifier: The name of the cluster.
        :param node_type: The type of node in the cluster.
        :param master_username: The master username.
        :param master_user_password: The master user password.
        :param publicly_accessible: Whether the cluster is publicly accessible.
        :param number_of_nodes: The number of nodes in the cluster.
        :return: The cluster.
        """

        try:
            cluster = self.client.create_cluster(
                ClusterIdentifier=cluster_identifier,
                NodeType=node_type,
                MasterUsername=master_username,
                MasterUserPassword=master_user_password,
                PubliclyAccessible=publicly_accessible,
                NumberOfNodes=number_of_nodes,
            )
            return cluster
        except ClientError as err:
```

```
            logging.error(
                "Couldn't create a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_clusters(self, cluster_identifier):
        """
        Describes a cluster.

        :param cluster_identifier: The cluster identifier.
        :return: A list of clusters.
        """
        try:
            kwargs = {}
            if cluster_identifier:
                kwargs["ClusterIdentifier"] = cluster_identifier

            paginator = self.client.get_paginator("describe_clusters")
            clusters = []
            for page in paginator.paginate(**kwargs):
                clusters.extend(page["Clusters"])

            return clusters

        except ClientError as err:
            logging.error(
                "Couldn't describe a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def execute_statement(
        self, cluster_identifier, database_name, user_name, sql,
parameter_list=None
    ):
        """
        Executes a SQL statement.

        :param cluster_identifier: The cluster identifier.
```

```
        :param database_name: The database name.
        :param user_name: The user's name.
        :param sql: The SQL statement.
        :param parameter_list: The optional SQL statement parameters.
        :return: The SQL statement result.
        """

        try:
            kwargs = {
                "ClusterIdentifier": cluster_identifier,
                "Database": database_name,
                "DbUser": user_name,
                "Sql": sql,
            }
            if parameter_list:
                kwargs["Parameters"] = parameter_list
            response = self.client.execute_statement(**kwargs)
            return response
        except ClientError as err:
            logging.error(
                "Couldn't execute statement. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_statement(self, statement_id):
        """
        Describes a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            response = self.client.describe_statement(Id=statement_id)
            return response
        except ClientError as err:
            logging.error(
                "Couldn't describe statement. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def get_statement_result(self, statement_id):
        """
        Gets the result of a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            result = {
                "Records": [],
            }
            paginator = self.client.get_paginator("get_statement_result")
            for page in paginator.paginate(Id=statement_id):
                if "ColumnMetadata" not in result:
                    result["ColumnMetadata"] = page["ColumnMetadata"]
                result["Records"].extend(page["Records"])
            return result
        except ClientError as err:
            logging.error(
                "Couldn't get statement result. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def modify_cluster(self, cluster_identifier, preferred_maintenance_window):
        """
        Modifies a cluster.

        :param cluster_identifier: The cluster identifier.
        :param preferred_maintenance_window: The preferred maintenance window.
        """
        try:
            self.client.modify_cluster(
                ClusterIdentifier=cluster_identifier,
                PreferredMaintenanceWindow=preferred_maintenance_window,
            )
        except ClientError as err:
            logging.error(
                "Couldn't modify a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise


    def list_databases(self, cluster_identifier, database_name, database_user):
        """
        Lists databases in a cluster.

        :param cluster_identifier: The cluster identifier.
        :param database_name: The database name.
        :param database_user: The database user.
        :return: The list of databases.
        """
        try:
            paginator = self.client.get_paginator("list_databases")
            databases = []
            for page in paginator.paginate(
                ClusterIdentifier=cluster_identifier,
                Database=database_name,
                DbUser=database_user,
            ):
                databases.extend(page["Databases"])

            return databases
        except ClientError as err:
            logging.error(
                "Couldn't list databases. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_cluster(self, cluster_identifier):
        """
        Deletes a cluster.

        :param cluster_identifier: The cluster identifier.
        """
        try:
            self.client.delete_cluster(
                ClusterIdentifier=cluster_identifier,
SkipFinalClusterSnapshot=True
```

```
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Actions for Amazon Redshift using AWS SDKs

The following code examples demonstrate how to perform individual Amazon Redshift actions with AWS SDKs. Each example includes a link to GitHub, where you can find instructions for setting up and running the code.

These excerpts call the Amazon Redshift API and are code excerpts from larger programs that must be run in context. You can see actions in context in Scenarios for Amazon Redshift using AWS SDKs.

The following examples include only the most commonly used actions. For a complete list, see the Amazon Redshift API Reference.

Examples

- Use CreateCluster with an AWS SDK or CLI

- Use DeleteCluster with an AWS SDK or CLI
- Use DescribeClusters with an AWS SDK or CLI
- Use DescribeStatement with an AWS SDK
- Use ExecuteStatement with an AWS SDK
- Use GetStatementResult with an AWS SDK
- Use ListDatabases with an AWS SDK
- Use ModifyCluster with an AWS SDK or CLI

Use `CreateCluster` with an AWS SDK or CLI

The following code examples show how to use `CreateCluster`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Learn the basics
- Getting started with provisioned data warehouse clusters

.NET

SDK for .NET (v4)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/// <summary>
/// Create a new Amazon Redshift cluster.
/// </summary>
/// <param name="clusterIdentifier">The identifier for the cluster.</param>
/// <param name="databaseName">The name of the database.</param>
/// <param name="masterUsername">The master username.</param>
/// <param name="masterUserPassword">The master user password.</param>
/// <param name="nodeType">The node type for the cluster.</param>
/// <returns>The cluster that was created.</returns>
```

```
   public async Task<Cluster> CreateClusterAsync(string clusterIdentifier,
string databaseName,
       string masterUsername, string masterUserPassword, string nodeType =
"ra3.large")
   {
       try
       {
           var request = new CreateClusterRequest
           {
               ClusterIdentifier = clusterIdentifier,
               DBName = databaseName,
               MasterUsername = masterUsername,
               MasterUserPassword = masterUserPassword,
               NodeType = nodeType,
               NumberOfNodes = 1,
               ClusterType = "single-node"
           };

           var response = await _redshiftClient.CreateClusterAsync(request);
           Console.WriteLine($"Created cluster {clusterIdentifier}");
           return response.Cluster;
       }
       catch (ClusterAlreadyExistsException ex)
       {
           Console.WriteLine($"Cluster already exists: {ex.Message}");
           throw;
       }
       catch (Exception ex)
       {
           Console.WriteLine($"Couldn't create cluster. Here's why:
{ex.Message}");
           throw;
       }
   }
```

- For API details, see CreateCluster in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To create a cluster with minimal parameters

The following `create-cluster` example creates a cluster with the minimal set of parameters. By default, the output is in JSON format.

```
aws redshift create-cluster \
    --node-type dw.hs1.xlarge \
    --number-of-nodes 2 \
    --master-username adminuser \
    --master-user-password TopSecret1 \
    --cluster-identifier mycluster
```

Output:

```
{
    "Cluster": {
        "NodeType": "dw.hs1.xlarge",
        "ClusterVersion": "1.0",
        "PubliclyAccessible": true,
        "MasterUsername": "adminuser",
        "ClusterParameterGroups": [
            {
                "ParameterApplyStatus": "in-sync",
                "ParameterGroupName": "default.redshift-1.0"
        } ],
        "ClusterSecurityGroups": [
            {
                "Status": "active",
                "ClusterSecurityGroupName": "default"
        } ],
        "AllowVersionUpgrade": true,
        "VpcSecurityGroups": [],
        "PreferredMaintenanceWindow": "sat:03:30-sat:04:00",
        "AutomatedSnapshotRetentionPeriod": 1,
        "ClusterStatus": "creating",
        "ClusterIdentifier": "mycluster",
        "DBName": "dev",
        "NumberOfNodes": 2,
        "PendingModifiedValues": {
            "MasterUserPassword": "****"
        }
    }
}
```

For more information, see Creating a cluster in the *Amazon Redshift Management Guide*.

- For API details, see CreateCluster in *AWS CLI Command Reference*.

Go

SDK for Go V2

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// CreateCluster sends a request to create a cluster with the given clusterId
 using the provided credentials.
func (actor RedshiftActions) CreateCluster(ctx context.Context, clusterId string,
 userName string, userPassword string, nodeType string, clusterType string,
 publiclyAccessible bool) (*redshift.CreateClusterOutput, error) {
 // Create a new Redshift cluster
 input := &redshift.CreateClusterInput{
```

```
  ClusterIdentifier:  aws.String(clusterId),
  MasterUserPassword: aws.String(userPassword),
  MasterUsername:     aws.String(userName),
  NodeType:           aws.String(nodeType),
  ClusterType:        aws.String(clusterType),
  PubliclyAccessible: aws.Bool(publiclyAccessible),
 }
 var opErr *types.ClusterAlreadyExistsFault
 output, err := actor.RedshiftClient.CreateCluster(ctx, input)
 if err != nil && errors.As(err, &opErr) {
  log.Println("Cluster already exists")
  return nil, nil
 } else if err != nil {
  log.Printf("Failed to create Redshift cluster: %v\n", err)
  return nil, err
 }

 log.Printf("Created cluster %s\n", *output.Cluster.ClusterIdentifier)
 return output, nil
}
```

- For API details, see CreateCluster in *AWS SDK for Go API Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create the cluster.

```
    /**
     * Creates a new Amazon Redshift cluster asynchronously.
     * @param clusterId     the unique identifier for the cluster
     * @param username      the username for the administrative user
     * @param userPassword  the password for the administrative user
```

```
     * @return a CompletableFuture that represents the asynchronous operation of
creating the cluster
     * @throws RuntimeException if the cluster creation fails
     */
    public CompletableFuture<CreateClusterResponse> createClusterAsync(String
clusterId, String username, String userPassword) {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(username)
            .masterUserPassword(userPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        return getAsyncClient().createCluster(clusterRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Created cluster ");
                } else {
                    throw new RuntimeException("Failed to create cluster: " +
exception.getMessage(), exception);
                }
            });
    }
```

- For API details, see CreateCluster in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create the client.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
```

```
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Create the cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { CreateClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
  ClusterIdentifier: "CLUSTER_NAME", // Required
  NodeType: "NODE_TYPE", //Required
  MasterUsername: "MASTER_USER_NAME", // Required - must be lowercase
  MasterUserPassword: "MASTER_USER_PASSWORD", // Required - must contain at least
 one uppercase letter, and one number
  ClusterType: "CLUSTER_TYPE", // Required
  IAMRoleARN: "IAM_ROLE_ARN", // Optional - the ARN of an IAM role with
 permissions your cluster needs to access other AWS services on your behalf, such
 as Amazon S3.
  ClusterSubnetGroupName: "CLUSTER_SUBNET_GROUPNAME", //Optional - the name of a
 cluster subnet group to be associated with this cluster. Defaults to 'default'
 if not specified.
  DBName: "DATABASE_NAME", // Optional - defaults to 'dev' if not specified
  Port: "PORT_NUMBER", // Optional - defaults to '5439' if not specified
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new CreateClusterCommand(params));
    console.log(
      `Cluster ${data.Cluster.ClusterIdentifier} successfully created`,
    );
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see CreateCluster in *AWS SDK for JavaScript API Reference*.

Kotlin

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create the cluster.

```
suspend fun createCluster(
    clusterId: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val clusterRequest =
        CreateClusterRequest {
            clusterIdentifier = clusterId
            availabilityZone = "us-east-1a"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
            nodeType = "ra3.4xlarge"
            publiclyAccessible = true
            numberOfNodes = 2
        }

    RedshiftClient.fromEnvironment { region = "us-east-1" }.use { redshiftClient
 ->
        val clusterResponse = redshiftClient.createCluster(clusterRequest)
        println("Created cluster ${clusterResponse.cluster?.clusterIdentifier}")
    }
}
```

- For API details, see CreateCluster in *AWS SDK for Kotlin API reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def create_cluster(
        self,
        cluster_identifier,
        node_type,
        master_username,
        master_user_password,
        publicly_accessible,
        number_of_nodes,
    ):
        """
        Creates a cluster.

        :param cluster_identifier: The name of the cluster.
        :param node_type: The type of node in the cluster.
        :param master_username: The master username.
        :param master_user_password: The master user password.
        :param publicly_accessible: Whether the cluster is publicly accessible.
        :param number_of_nodes: The number of nodes in the cluster.
        :return: The cluster.
        """
```

```
        try:
            cluster = self.client.create_cluster(
                ClusterIdentifier=cluster_identifier,
                NodeType=node_type,
                MasterUsername=master_username,
                MasterUserPassword=master_user_password,
                PubliclyAccessible=publicly_accessible,
                NumberOfNodes=number_of_nodes,
            )
            return cluster
        except ClientError as err:
            logging.error(
                "Couldn't create a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

The following code instantiates the RedshiftWrapper object.

```
client = boto3.client("redshift")
redhift_wrapper = RedshiftWrapper(client)
```

- For API details, see CreateCluster in *AWS SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create the cluster.

```
    TRY.
        " Example values: iv_cluster_identifier = 'my-redshift-cluster'
        " Example values: iv_node_type = 'ra3.4xlarge'
        " Example values: iv_master_username = 'awsuser'
        " Example values: iv_master_password = 'AwsUser1000'
        " Example values: iv_publicly_accessible = abap_true
        " Example values: iv_number_of_nodes = 2
        oo_result = lo_rsh->createcluster(
          iv_clusteridentifier = iv_cluster_identifier
          iv_nodetype = iv_node_type
          iv_masterusername = iv_master_username
          iv_masteruserpassword = iv_master_password
          iv_publiclyaccessible = iv_publicly_accessible
          iv_numberofnodes = iv_number_of_nodes
        ).
        MESSAGE 'Redshift cluster created successfully.' TYPE 'I'.
      CATCH /aws1/cx_rshclustalrdyexfault.
        MESSAGE 'Cluster already exists.' TYPE 'I'.
      CATCH /aws1/cx_rshclstquotaexcdfault.
        MESSAGE 'Cluster quota exceeded.' TYPE 'I'.
    ENDTRY.
```

- For API details, see CreateCluster in *AWS SDK for SAP ABAP API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteCluster` with an AWS SDK or CLI

The following code examples show how to use `DeleteCluster`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Getting started with provisioned data warehouse clusters

.NET

SDK for .NET (v4)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   /// <summary>
   /// Delete an Amazon Redshift cluster without a final snapshot.
   /// </summary>
   /// <param name="clusterIdentifier">The identifier for the cluster.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> DeleteClusterWithoutSnapshotAsync(string
clusterIdentifier)
   {
       try
       {
           var request = new DeleteClusterRequest
           {
               ClusterIdentifier = clusterIdentifier,
               SkipFinalClusterSnapshot = true
           };

           var response = await _redshiftClient.DeleteClusterAsync(request);
           Console.WriteLine($"The {clusterIdentifier} was deleted");
           return true;
       }
       catch (ClusterNotFoundException ex)
       {
           Console.WriteLine($"Cluster not found: {ex.Message}");
           return false;
       }
       catch (Exception ex)
       {
           Console.WriteLine($"Couldn't delete cluster. Here's why:
{ex.Message}");
           return false;
       }
   }
```

- For API details, see DeleteCluster in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

Example 1: To delete a cluster with no final cluster snapshot

The following `delete-cluster` example deletes a cluster, forcing data deletion so no final cluster snapshot is created.

```
aws redshift delete-cluster \
    --cluster-identifier mycluster \
    --skip-final-cluster-snapshot
```

Output:

```
{
    "Cluster": {
        "ClusterIdentifier": "mycluster",
        "NodeType": "dc2.large",
        "ClusterStatus": "deleting",
        "ClusterAvailabilityStatus": "Modifying",
        "MasterUsername": "adminuser",
        "DBName": "dev",
        "Endpoint": {
            "Address": "mycluster.c1a2b3c4d5e6.us-west-2.redshift.amazonaws.com",
            "Port": 5439
        },
        "ClusterCreateTime": "2019-12-23T21:42:51.430Z",
        "AutomatedSnapshotRetentionPeriod": 1,
        "ManualSnapshotRetentionPeriod": -1,
        "ClusterSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-0a1b2c3d4e5f6a7b8",
                "Status": "active"
            }
        ],
        "ClusterParameterGroups": [
```

```
            {
                "ParameterGroupName": "default.redshift-1.0",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "ClusterSubnetGroupName": "default",
        "VpcId": "vpc-0a1b2c3d4e5f6a7b8",
        "AvailabilityZone": "us-west-2a",
        "PreferredMaintenanceWindow": "sat:03:30-sat:04:00",
        "PendingModifiedValues": {},
        "ClusterVersion": "1.0",
        "AllowVersionUpgrade": true,
        "NumberOfNodes": 2,
        "PubliclyAccessible": false,
        "Encrypted": false,
        "Tags": [],
        "EnhancedVpcRouting": false,
        "IamRoles": [],
        "MaintenanceTrackName": "current",
        "DeferredMaintenanceWindows": [],
        "NextMaintenanceWindowStartTime": "2019-12-28T03:30:00Z"
    }
}
```

For more information, see Shutting down and deleting a cluster in the *Amazon Redshift Management Guide*.

Example 2: To delete a cluster, allowing a final cluster snapshot

The following `delete-cluster` example deletes a cluster, but specifies a final cluster snapshot.

```
aws redshift delete-cluster \
    --cluster-identifier mycluster \
    --final-cluster-snapshot-identifier myfinalsnapshot
```

Output:

```
{
    "Cluster": {
        "ClusterIdentifier": "mycluster",
        "NodeType": "dc2.large",
        "ClusterStatus": "deleting",
```

```
        "ClusterAvailabilityStatus": "Modifying",
        "MasterUsername": "adminuser",
        "DBName": "dev",
        "Endpoint": {
            "Address": "mycluster.c1a2b3c4d5e6.us-west-2.redshift.amazonaws.com",
            "Port": 5439
        },
        "ClusterCreateTime": "2019-12-23T21:42:51.430Z",
        "AutomatedSnapshotRetentionPeriod": 1,
        "ManualSnapshotRetentionPeriod": -1,
        "ClusterSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-0a1b2c3d4e5f6a7b8",
                "Status": "active"
            }
        ],
        "ClusterParameterGroups": [
            {
                "ParameterGroupName": "default.redshift-1.0",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "ClusterSubnetGroupName": "default",
        "VpcId": "vpc-0a1b2c3d4e5f6a7b8",
        "AvailabilityZone": "us-west-2a",
        "PreferredMaintenanceWindow": "sat:03:30-sat:04:00",
        "PendingModifiedValues": {},
        "ClusterVersion": "1.0",
        "AllowVersionUpgrade": true,
        "NumberOfNodes": 2,
        "PubliclyAccessible": false,
        "Encrypted": false,
        "Tags": [],
        "EnhancedVpcRouting": false,
        "IamRoles": [],
        "MaintenanceTrackName": "current",
        "DeferredMaintenanceWindows": [],
        "NextMaintenanceWindowStartTime": "2019-12-28T03:30:00Z"
    }
}
```

For more information, see Shutting down and deleting a cluster in the *Amazon Redshift Management Guide*.

- For API details, see DeleteCluster in *AWS CLI Command Reference*.

Go

SDK for Go V2

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// DeleteCluster deletes the given cluster.
func (actor RedshiftActions) DeleteCluster(ctx context.Context, clusterId string)
 (bool, error) {
 input := redshift.DeleteClusterInput{
  ClusterIdentifier:        aws.String(clusterId),
  SkipFinalClusterSnapshot: aws.Bool(true),
```

```
 }
 _, err := actor.RedshiftClient.DeleteCluster(ctx, &input)
 var opErr *types.ClusterNotFoundFault
 if err != nil && errors.As(err, &opErr) {
  log.Println("Cluster was not found. Where could it be?")
  return false, err
 } else if err != nil {
  log.Printf("Failed to delete Redshift cluster: %v\n", err)
  return false, err
 }
 waiter := redshift.NewClusterDeletedWaiter(actor.RedshiftClient)
 err = waiter.Wait(ctx, &redshift.DescribeClustersInput{
  ClusterIdentifier: aws.String(clusterId),
 }, 5*time.Minute)
 if err != nil {
  log.Printf("Wait time exceeded for deleting cluster, continuing: %v\n", err)
 }
 log.Printf("The cluster %s was deleted\n", clusterId)
 return true, nil
}
```

- For API details, see DeleteCluster in *AWS SDK for Go API Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete the cluster.

```
    /**
     * Deletes a Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the Redshift cluster to be deleted
```

```
    * @return a {@link CompletableFuture} that represents the asynchronous
operation of deleting the Redshift cluster
    */
   public CompletableFuture<DeleteClusterResponse>
deleteRedshiftClusterAsync(String clusterId) {
       DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .skipFinalClusterSnapshot(true)
           .build();

       return getAsyncClient().deleteCluster(deleteClusterRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   // Handle exceptions
                   if (exception.getCause() instanceof RedshiftException) {
                       logger.info("Error: {}", exception.getMessage());
                   } else {
                       logger.info("Unexpected error: {}",
exception.getMessage());
                   }
               } else {
                   // Handle successful response
                   logger.info("The status is {}",
response.cluster().clusterStatus());
               }
           });
   }
```

- For API details, see DeleteCluster in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create the client.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Create the cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { DeleteClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
  ClusterIdentifier: "CLUSTER_NAME",
  SkipFinalClusterSnapshot: false,
  FinalClusterSnapshotIdentifier: "CLUSTER_SNAPSHOT_ID",
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new DeleteClusterCommand(params));
    console.log("Success, cluster deleted. ", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see DeleteCluster in *AWS SDK for JavaScript API Reference*.

Kotlin

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete the cluster.

```
suspend fun deleteRedshiftCluster(clusterId: String?) {
    val request =
        DeleteClusterRequest {
            clusterIdentifier = clusterId
            skipFinalClusterSnapshot = true
        }

    RedshiftClient.fromEnvironment { region = "us-west-2" }.use { redshiftClient
 ->
        val response = redshiftClient.deleteCluster(request)
        println("The status is ${response.cluster?.clusterStatus}")
    }
}
```

- For API details, see DeleteCluster in *AWS SDK for Kotlin API reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class RedshiftWrapper:
```

```
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def delete_cluster(self, cluster_identifier):
        """
        Deletes a cluster.

        :param cluster_identifier: The cluster identifier.
        """
        try:
            self.client.delete_cluster(
                ClusterIdentifier=cluster_identifier,
SkipFinalClusterSnapshot=True
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

The following code instantiates the RedshiftWrapper object.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- For API details, see DeleteCluster in *AWS SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete the cluster.

```
TRY.
    " Example values: iv_cluster_identifier = 'my-redshift-cluster'
    lo_rsh->deletecluster(
      iv_clusteridentifier = iv_cluster_identifier
      iv_skipfinalclustersnapshot = abap_true
    ).
    MESSAGE 'Redshift cluster deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_rshclustnotfoundfault.
    MESSAGE 'Cluster not found.' TYPE 'I'.
  CATCH /aws1/cx_rshinvcluststatefault.
    MESSAGE 'Invalid cluster state for deletion.' TYPE 'I'.
ENDTRY.
```

- For API details, see DeleteCluster in *AWS SDK for SAP ABAP API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeClusters` with an AWS SDK or CLI

The following code examples show how to use `DescribeClusters`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Learn the basics
- Getting started with provisioned data warehouse clusters

.NET

SDK for .NET (v4)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   /// <summary>
   /// Describe Amazon Redshift clusters.
   /// </summary>
   /// <param name="clusterIdentifier">Optional cluster identifier to describe a
specific cluster.</param>
   /// <returns>A list of clusters.</returns>
   public async Task<List<Cluster>> DescribeClustersAsync(string?
clusterIdentifier = null)
   {
       try
       {
           var clusters = new List<Cluster>();
           var request = new DescribeClustersRequest();
           if (!string.IsNullOrEmpty(clusterIdentifier))
           {
               request.ClusterIdentifier = clusterIdentifier;
           }

           var clustersPaginator =
_redshiftClient.Paginators.DescribeClusters(request);
           await foreach (var response in clustersPaginator.Responses)
           {
               if (response.Clusters != null)
                   clusters.AddRange(response.Clusters);
           }

           Console.WriteLine($"{clusters.Count} cluster(s) retrieved.");
           foreach (var cluster in clusters)
           {
               Console.WriteLine($"\t{cluster.ClusterIdentifier} (Status:
{cluster.ClusterStatus})");
           }
```

```
            return clusters;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
 {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe clusters. Here's why:
 {ex.Message}");
            throw;
        }
    }
```

- For API details, see DescribeClusters in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To get a description of all clusters

The following `describe-clusters` example returns a description of all clusters for the account. By default, the output is in JSON format.

```
aws redshift describe-clusters
```

Output:

```
{
    "Clusters": [
        {
            "NodeType": "dw.hs1.xlarge",
            "Endpoint": {
                "Port": 5439,
                "Address": "mycluster.coqoarplqhsn.us-
west-2.redshift.amazonaws.com"
            },
```

```
            "ClusterVersion": "1.0",
            "PubliclyAccessible": true,
            "MasterUsername": "adminuser",
            "ClusterParameterGroups": [
                {
                    "ParameterApplyStatus": "in-sync",
                    "ParameterGroupName": "default.redshift-1.0"
            } ],
            "ClusterSecurityGroups": [
                {
                    "Status": "active",
                    "ClusterSecurityGroupName": "default"
            } ],
            "AllowVersionUpgrade": true,
            "VpcSecurityGroups": [],
            "AvailabilityZone": "us-west-2a",
            "ClusterCreateTime": "2013-01-22T21:59:29.559Z",
            "PreferredMaintenanceWindow": "sat:03:30-sat:04:00",
            "AutomatedSnapshotRetentionPeriod": 1,
            "ClusterStatus": "available",
            "ClusterIdentifier": "mycluster",
            "DBName": "dev",
            "NumberOfNodes": 2,
            "PendingModifiedValues": {}
    } ]
}
```

You can also obtain the same information in text format using the `--output text` option.

```
aws redshift describe-clusters \
    --output text
```

Output:

```
dw.hs1.xlarge       1.0     true    adminuser       True    us-west-2a
 2013-01-22T21:59:29.559Z        sat:03:30-sat:04:00     1       available
 mycluster       dev     2
ENDPOINT    5439    mycluster.coqoarplqhsn.us-west-2.redshift.amazonaws.com
in-sync     default.redshift-1.0
active      default
PENDINGMODIFIEDVALUES
```

For more information, see Amazon Redshift provisioned clusters in the *Amazon Redshift Management Guide*.

- For API details, see DescribeClusters in *AWS CLI Command Reference*.

Go

SDK for Go V2

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// DescribeClusters returns information about the given cluster.
func (actor RedshiftActions) DescribeClusters(ctx context.Context, clusterId
 string) (*redshift.DescribeClustersOutput, error) {
 input, err := actor.RedshiftClient.DescribeClusters(ctx,
 &redshift.DescribeClustersInput{
  ClusterIdentifier: aws.String(clusterId),
```

```
 })
 var opErr *types.AccessToClusterDeniedFault
 if errors.As(err, &opErr) {
  println("Access to cluster denied.")
  panic(err)
 } else if err != nil {
  println("Failed to describe Redshift clusters.")
  return nil, err
 }
 return input, nil
}
```

- For API details, see DescribeClusters in *AWS SDK for Go API Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Describe the cluster.

```
    /**
     * Waits asynchronously for the specified cluster to become available.
     * @param clusterId the identifier of the cluster to wait for
     * @return a {@link CompletableFuture} that completes when the cluster is
ready
     */
    public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
        DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();

        logger.info("Waiting for cluster to become available. This may take a few
minutes.");
```

```
        long startTime = System.currentTimeMillis();

        // Recursive method to poll the cluster status.
        return checkClusterStatusAsync(clustersRequest, startTime);
    }

    private CompletableFuture<Void>
checkClusterStatusAsync(DescribeClustersRequest clustersRequest, long startTime)
{
        return getAsyncClient().describeClusters(clustersRequest)
            .thenCompose(clusterResponse -> {
                List<Cluster> clusterList = clusterResponse.clusters();
                boolean clusterReady = false;
                for (Cluster cluster : clusterList) {
                    if ("available".equals(cluster.clusterStatus())) {
                        clusterReady = true;
                        break;
                    }
                }

                if (clusterReady) {
                    logger.info(String.format("Cluster is available!"));
                    return CompletableFuture.completedFuture(null);
                } else {
                    long elapsedTimeMillis = System.currentTimeMillis() -
startTime;
                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;
                    System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
                    System.out.flush();

                    // Wait 1 second before the next status check
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
                }
```

```
            }).exceptionally(exception -> {
                throw new RuntimeException("Failed to get cluster status: " +
 exception.getMessage(), exception);
            });
    }
```

- For API details, see DescribeClusters in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create the client.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Describe your clusters.

```
// Import required AWS SDK clients and commands for Node.js
import { DescribeClustersCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
  ClusterIdentifier: "CLUSTER_NAME",
};

const run = async () => {
  try {
```

```
    const data = await redshiftClient.send(new DescribeClustersCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see DescribeClusters in *AWS SDK for JavaScript API Reference*.

Kotlin

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Describe the cluster.

```
suspend fun describeRedshiftClusters() {
    RedshiftClient.fromEnvironment { region = "us-west-2" }.use { redshiftClient
 ->
        val clusterResponse =
 redshiftClient.describeClusters(DescribeClustersRequest {})
        val clusterList = clusterResponse.clusters

        if (clusterList != null) {
            for (cluster in clusterList) {
                println("Cluster database name is ${cluster.dbName}")
                println("Cluster status is ${cluster.clusterStatus}")
            }
        }
    }
}
```

- For API details, see DescribeClusters in *AWS SDK for Kotlin API reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def describe_clusters(self, cluster_identifier):
        """
        Describes a cluster.

        :param cluster_identifier: The cluster identifier.
        :return: A list of clusters.
        """
        try:
            kwargs = {}
            if cluster_identifier:
                kwargs["ClusterIdentifier"] = cluster_identifier

            paginator = self.client.get_paginator("describe_clusters")
            clusters = []
            for page in paginator.paginate(**kwargs):
                clusters.extend(page["Clusters"])

            return clusters

        except ClientError as err:
```

```
            logging.error(
                "Couldn't describe a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

The following code instantiates the RedshiftWrapper object.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- For API details, see DescribeClusters in *AWS SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Describe the cluster.

```
    TRY.
        " Example values: iv_cluster_identifier = 'my-redshift-
cluster' (optional)
        oo_result = lo_rsh->describeclusters(
          iv_clusteridentifier = iv_cluster_identifier
        ).
        lt_clusters = oo_result->get_clusters( ).
        lv_cluster_count = lines( lt_clusters ).
        MESSAGE |Retrieved { lv_cluster_count } cluster(s).| TYPE 'I'.
      CATCH /aws1/cx_rshclustnotfoundfault.
        MESSAGE 'Cluster not found.' TYPE 'I'.
    ENDTRY.
```

- For API details, see DescribeClusters in *AWS SDK for SAP ABAP API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeStatement` with an AWS SDK

The following code examples show how to use `DescribeStatement`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Learn the basics

.NET

SDK for .NET (v4)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /// <summary>
    /// Describe a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>The statement description.</returns>
    public async Task<DescribeStatementResponse> DescribeStatementAsync(string
statementId)
    {
        try
        {
            var request = new DescribeStatementRequest
            {
                Id = statementId
```

```
            };

            var response = await
_redshiftDataClient.DescribeStatementAsync(request);
            return response;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe statement. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- For API details, see DescribeStatement in *AWS SDK for .NET API Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /**
     * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
     *
     * @param sqlId the ID of the SQL statement to check
     * @return a {@link CompletableFuture} that completes when the SQL
statement's status is either "FINISHED" or "FAILED"
     */
    public CompletableFuture<Void> checkStatementAsync(String sqlId) {
```

```
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();

        return getAsyncDataClient().describeStatement(statementRequest)
            .thenCompose(response -> {
                String status = response.statusAsString();
                logger.info("... Status: {} ", status);

                if ("FAILED".equals(status)) {
                    throw new RuntimeException("The Query Failed. Ending
program");
                } else if ("FINISHED".equals(status)) {
                    return CompletableFuture.completedFuture(null);
                } else {
                    // Sleep for 1 second and recheck status
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
Recursively call until status is FINISHED or FAILED
                }
            }).whenComplete((result, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    logger.info("Error: {} ", exception.getMessage());
                } else {
                    logger.info("The statement is finished!");
                }
            });
    }
```

- For API details, see DescribeStatement in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class RedshiftDataWrapper:
    """Encapsulates Amazon Redshift data."""

    def __init__(self, client):
        """
        :param client: A Boto3 RedshiftDataWrapper client.
        """
        self.client = client


    def describe_statement(self, statement_id):
        """
        Describes a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            response = self.client.describe_statement(Id=statement_id)
            return response
        except ClientError as err:
            logging.error(
                "Couldn't describe statement. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

The following code instantiates the RedshiftDataWrapper object.

```
    client = boto3.client("redshift-data")
    redshift_data_wrapper = RedshiftDataWrapper(client)
```

- For API details, see DescribeStatement in *AWS SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    TRY.
        " Example values: iv_statement_id = 'xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx'
        oo_result = lo_rsd->describestatement(
          iv_id = iv_statement_id
        ).
        lv_status = oo_result->get_status( ).
        MESSAGE |Statement status: { lv_status }| TYPE 'I'.
      CATCH /aws1/cx_rsdresourcenotfoundex.
        MESSAGE 'Statement not found.' TYPE 'I'.
      CATCH /aws1/cx_rsdinternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
    ENDTRY.
```

- For API details, see DescribeStatement in *AWS SDK for SAP ABAP API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ExecuteStatement` with an AWS SDK

The following code examples show how to use `ExecuteStatement`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Learn the basics

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Executes a SQL statement to create a database table.

```
   /**
    * Creates an asynchronous task to execute a SQL statement for creating a new
table.
    *
    * @param clusterId    the identifier of the Amazon Redshift cluster
    * @param databaseName the name of the database to create the table in
    * @param userName     the username to use for the database connection
    * @return a {@link CompletableFuture} that completes with the result of the
SQL statement execution
    * @throws RuntimeException if there is an error creating the table
    */
   public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
       ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
           .clusterIdentifier(clusterId)
           .dbUser(userName)
           .database(databaseName)
           .sql("CREATE TABLE Movies (" +
               "id INT PRIMARY KEY, " +
               "title VARCHAR(100), " +
               "year INT)")
           .build();

       return getAsyncDataClient().executeStatement(createTableRequest)
```

```
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
                } else {
                    logger.info("Table created: Movies");
                }
            });
    }
```

Executes a SQL statement to insert data into a database table.

```
    /**
     * Asynchronously pops a table from a JSON file.
     *
     * @param clusterId   the ID of the cluster
     * @param databaseName the name of the database
     * @param userName    the username
     * @param fileName    the name of the JSON file
     * @param number      the number of records to process
     * @return a CompletableFuture that completes with the number of records
added to the Movies table
     */
    public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
        return CompletableFuture.supplyAsync(() -> {
                try {
                    JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                    JsonNode rootNode = new ObjectMapper().readTree(parser);
                    Iterator<JsonNode> iter = rootNode.iterator();
                    return iter;
                } catch (IOException e) {
                    throw new RuntimeException("Failed to read or parse JSON
file: " + e.getMessage(), e);
                }
            }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    logger.info("Error {} ", exception.getMessage());
                } else {
```

```
                    logger.info("{} records were added to the Movies table." ,
result);
                }
            });
    }

    private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
        return CompletableFuture.supplyAsync(() -> {
            int t = 0;
            try {
                while (iter.hasNext()) {
                    if (t == number)
                        break;
                    JsonNode currentNode = iter.next();
                    int year = currentNode.get("year").asInt();
                    String title = currentNode.get("title").asText();

                    // Use SqlParameter to avoid SQL injection.
                    List<SqlParameter> parameterList = new ArrayList<>();
                    String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";
                    SqlParameter idParam = SqlParameter.builder()
                        .name("id")
                        .value(String.valueOf(t))
                        .build();

                    SqlParameter titleParam = SqlParameter.builder()
                        .name("title")
                        .value(title)
                        .build();

                    SqlParameter yearParam = SqlParameter.builder()
                        .name("year")
                        .value(String.valueOf(year))
                        .build();
                    parameterList.add(idParam);
                    parameterList.add(titleParam);
                    parameterList.add(yearParam);

                    ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                        .clusterIdentifier(clusterId)
                        .sql(sqlStatement)
```

```
                        .database(databaseName)
                        .dbUser(userName)
                        .parameters(parameterList)
                        .build();


getAsyncDataClient().executeStatement(insertStatementRequest);
                    logger.info("Inserted: " + title + " (" + year + ")");
                    t++;
                }
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
            }
            return t;
        });
    }
```

Executes a SQL statement to query a database table.

```
    /**
     * Asynchronously queries movies by a given year from a Redshift database.
     *
     * @param database    the name of the database to query
     * @param dbUser      the user to connect to the database with
     * @param year        the year to filter the movies by
     * @param clusterId   the identifier of the Redshift cluster to connect to
     * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
     */
    public CompletableFuture<String> queryMoviesByYearAsync(String database,
                                                            String dbUser,
                                                            int year,
                                                            String
clusterId) {

        String sqlStatement = "SELECT * FROM Movies WHERE year = :year";
        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();
```

```
        ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .database(database)
            .dbUser(dbUser)
            .parameters(yearParam)
            .sql(sqlStatement)
            .build();

        return CompletableFuture.supplyAsync(() -> {
            try {
                ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
                return response.id();
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
            }
        }).exceptionally(exception -> {
            logger.info("Error: {}", exception.getMessage());
            return "";
        });
    }
```

- For API details, see ExecuteStatement in *AWS SDK for Java 2.x API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    TRY.
        " Example values: iv_cluster_identifier = 'redshift-cluster-movies'
        " Example values: iv_database_name = 'dev'
        " Example values: iv_user_name = 'awsuser'
```

```
        " Example values: iv_sql = 'SELECT * FROM movies WHERE year = :year'
        " Example values: it_parameter_list - SQL parameters for parameterized
queries

        " Only pass parameters if the list is not empty
        IF it_parameter_list IS NOT INITIAL.
          oo_result = lo_rsd->executestatement(
            iv_clusteridentifier = iv_cluster_identifier
            iv_database = iv_database_name
            iv_dbuser = iv_user_name
            iv_sql = iv_sql
            it_parameters = it_parameter_list
          ).
        ELSE.
          oo_result = lo_rsd->executestatement(
            iv_clusteridentifier = iv_cluster_identifier
            iv_database = iv_database_name
            iv_dbuser = iv_user_name
            iv_sql = iv_sql
          ).
        ENDIF.

        lv_statement_id = oo_result->get_id( ).
        MESSAGE |Statement executed. ID: { lv_statement_id }| TYPE 'I'.
      CATCH /aws1/cx_rsdexecutestatementex.
        MESSAGE 'Statement execution error.' TYPE 'I'.
      CATCH /aws1/cx_rsdresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
    ENDTRY.
```

- For API details, see ExecuteStatement in *AWS SDK for SAP ABAP API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetStatementResult` with an AWS SDK

The following code examples show how to use `GetStatementResult`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Learn the basics

.NET

SDK for .NET (v4)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   /// <summary>
   /// Get the results of a statement execution.
   /// </summary>
   /// <param name="statementId">The statement ID.</param>
   /// <returns>A list of result rows.</returns>
   public async Task<List<List<Field>>> GetStatementResultAsync(string
statementId)
   {
       try
       {
           var request = new GetStatementResultRequest
           {
               Id = statementId
           };

           var response = await
_redshiftDataClient.GetStatementResultAsync(request);
           return response.Records;
       }
       catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
       {
           Console.WriteLine($"Statement not found: {ex.Message}");
           throw;
       }
       catch (Exception ex)
       {
```

```
            Console.WriteLine($"Couldn't get statement result. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- For API details, see GetStatementResult in *AWS SDK for .NET API Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Check the statement result.

```
    /**
     * Asynchronously retrieves the results of a statement execution.
     *
     * @param statementId the ID of the statement for which to retrieve the
results
     * @return a {@link CompletableFuture} that completes when the statement
result has been processed
     */
    public CompletableFuture<Void> getResultsAsync(String statementId) {
        GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
            .id(statementId)
            .build();

        return getAsyncDataClient().getStatementResult(resultRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.info("Error getting statement result {} ",
exception.getMessage());
                    throw new RuntimeException("Error getting statement result: "
+ exception.getMessage(), exception);
```

```
                }

                // Extract and print the field values using streams if the
 response is valid.
                response.records().stream()
                    .flatMap(List::stream)
                    .map(Field::stringValue)
                    .filter(value -> value != null)
                    .forEach(value -> System.out.println("The Movie title field
 is " + value));

                return response;
            }).thenAccept(response -> {
                // Optionally add more logic here if needed after handling the
 response
            });
    }
```

- For API details, see GetStatementResult in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class RedshiftDataWrapper:
    """Encapsulates Amazon Redshift data."""

    def __init__(self, client):
        """
        :param client: A Boto3 RedshiftDataWrapper client.
        """
        self.client = client


    def get_statement_result(self, statement_id):
```

```
        """
        Gets the result of a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            result = {
                "Records": [],
            }
            paginator = self.client.get_paginator("get_statement_result")
            for page in paginator.paginate(Id=statement_id):
                if "ColumnMetadata" not in result:
                    result["ColumnMetadata"] = page["ColumnMetadata"]
                result["Records"].extend(page["Records"])
            return result
        except ClientError as err:
            logging.error(
                "Couldn't get statement result. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

The following code instantiates the RedshiftDataWrapper object.

```
    client = boto3.client("redshift-data")
    redshift_data_wrapper = RedshiftDataWrapper(client)
```

- For API details, see GetStatementResult in *AWS SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Check the statement result.

```
    TRY.
        " Example values: iv_statement_id = 'xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx'
        " Handle pagination for large result sets

        DO.
          lo_result_page = lo_rsd->getstatementresult(
            iv_id = iv_statement_id
            iv_nexttoken = lv_next_token
          ).

          " Collect records from this page
          lt_page_records = lo_result_page->get_records( ).
          APPEND LINES OF lt_page_records TO lt_all_records.

          " Check if there are more pages
          lv_next_token = lo_result_page->get_nexttoken( ).
          IF lv_next_token IS INITIAL.
            EXIT. " No more pages
          ENDIF.
        ENDDO.

        " For the last call, set oo_result for return value
        oo_result = lo_result_page.
        lv_record_count = lines( lt_all_records ).
        MESSAGE |Retrieved { lv_record_count } record(s).| TYPE 'I'.
      CATCH /aws1/cx_rsdresourcenotfoundex.
        MESSAGE 'Statement not found or results not available.' TYPE 'I'.
      CATCH /aws1/cx_rsdinternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
    ENDTRY.
```

- For API details, see GetStatementResult in *AWS SDK for SAP ABAP API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListDatabases` with an AWS SDK

The following code examples show how to use `ListDatabases`.

.NET

SDK for .NET (v4)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /// <summary>
    /// List databases in a Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="dbUser">The database name for authentication.</param>
    /// <returns>A list of database names.</returns>
    public async Task<List<string>> ListDatabasesAsync(string clusterIdentifier,
string dbUser, string databaseName)
    {
        try
        {
            var request = new ListDatabasesRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DbUser = dbUser,
                Database = databaseName
            };
```

```
            var response = await _redshiftDataClient.ListDatabasesAsync(request);
            var databases = new List<string>();

            foreach (var database in response.Databases)
            {
                Console.WriteLine($"The database name is : {database}");
                databases.Add(database);
            }

            return databases;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list databases. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- For API details, see ListDatabases in *AWS SDK for .NET API Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /**
     * Lists all databases asynchronously for the specified cluster, database
user, and database.
     * @param clusterId the identifier of the cluster to list databases for
     * @param dbUser the database user to use for the list databases request
```

```
     * @param database the database to list databases for
     * @return a {@link CompletableFuture} that completes when the database
listing is complete, or throws a {@link RuntimeException} if there was an error
     */
   public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
       ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
           .clusterIdentifier(clusterId)
           .dbUser(dbUser)
           .database(database)
           .build();

       // Asynchronous paginator for listing databases.
       ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
       CompletableFuture<Void> future = databasesPaginator.subscribe(response ->
{
           response.databases().forEach(db -> {
               logger.info("The database name is {} ", db);
           });
       });

       // Return the future for asynchronous handling.
       return future.exceptionally(exception -> {
           throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
       });
   }
```

- For API details, see ListDatabases in *AWS SDK for Java 2.x API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    TRY.
        " Example values: iv_cluster_identifier = 'redshift-cluster-movies'
        " Example values: iv_database_name = 'dev'
        " Example values: iv_database_user = 'awsuser'
        oo_result = lo_rsd->listdatabases(
          iv_clusteridentifier = iv_cluster_identifier
          iv_database = iv_database_name
          iv_dbuser = iv_database_user
        ).
        lt_databases = oo_result->get_databases( ).
        lv_db_count = lines( lt_databases ).
        MESSAGE |Retrieved { lv_db_count } database(s).| TYPE 'I'.
      CATCH /aws1/cx_rsddatabaseconnex.
        MESSAGE 'Database connection error.' TYPE 'I'.
      CATCH /aws1/cx_rsdresourcenotfoundex.
        MESSAGE 'Cluster not found.' TYPE 'I'.
    ENDTRY.
```

- For API details, see ListDatabases in *AWS SDK for SAP ABAP API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ModifyCluster` with an AWS SDK or CLI

The following code examples show how to use `ModifyCluster`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Learn the basics

.NET

SDK for .NET (v4)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   /// <summary>
   /// Modify an Amazon Redshift cluster.
   /// </summary>
   /// <param name="clusterIdentifier">The identifier for the cluster.</param>
   /// <param name="preferredMaintenanceWindow">The preferred maintenance
window.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> ModifyClusterAsync(string clusterIdentifier, string
preferredMaintenanceWindow)
   {
       try
       {
           var request = new ModifyClusterRequest
           {
               ClusterIdentifier = clusterIdentifier,
               PreferredMaintenanceWindow = preferredMaintenanceWindow
           };

           var response = await _redshiftClient.ModifyClusterAsync(request);
           Console.WriteLine($"The modified cluster was successfully modified
and has {response.Cluster.PreferredMaintenanceWindow} as the maintenance
window");
           return true;
       }
       catch (ClusterNotFoundException ex)
       {
           Console.WriteLine($"Cluster {clusterIdentifier} not found:
{ex.Message}");
           return false;
       }
       catch (Exception ex)
       {
```

```
            Console.WriteLine($"Couldn't modify cluster. Here's why:
 {ex.Message}");
            return false;
        }
    }
```

- For API details, see ModifyCluster in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

Example 1: To associate a security group with a cluster

The following `modify-cluster` example associates a cluster security group with the specified cluster.

```
aws redshift modify-cluster \
    --cluster-identifier mycluster \
    --cluster-security-groups mysecuritygroup
```

Output:

```
{
    "Cluster": {
        "ClusterIdentifier": "mycluster",
        "NodeType": "dc2.large",
        "ClusterStatus": "available",
        "ClusterAvailabilityStatus": "Available",
        "MasterUsername": "adminuser",
        "DBName": "dev",
        "Endpoint": {
            "Address": "mycluster.cmeaswqeuae.us-west-2.redshift.amazonaws.com",
            "Port": 5439
        },
        "ClusterCreateTime": "2019-12-02T22:45:07.890Z",
        "AutomatedSnapshotRetentionPeriod": 1,
        "ManualSnapshotRetentionPeriod": -1,
        "ClusterSecurityGroups": [
            {
                "ClusterSecurityGroupName": "default",
```

```
                "Status": "removing"
            },
            {
                "ClusterSecurityGroupName": "mysecuritygroup",
                "Status": "adding"
            }
        ],
        "VpcSecurityGroups": [],
        "ClusterParameterGroups": [
            {
                "ParameterGroupName": "default.redshift-1.0",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "AvailabilityZone": "us-west-2a",
        "PreferredMaintenanceWindow": "sat:03:30-sat:04:00",
        "PendingModifiedValues": {},
        "ClusterVersion": "1.0",
        "AllowVersionUpgrade": true,
        "NumberOfNodes": 2,
        "PubliclyAccessible": true,
        "Encrypted": false,
        "Tags": [],
        "EnhancedVpcRouting": false,
        "IamRoles": [],
        "MaintenanceTrackName": "current"
    }
}
```

For more information, see Modifying a cluster in the *Amazon Redshift Management Guide*.

Example 2: To modify the maintenance window for a cluster

The following `modify-cluster` example changes the weekly preferred maintenance window for a cluster to be the minimum four hour window starting Sundays at 11:15 PM, and ending Mondays at 3:15 AM.

```
aws redshift modify-cluster \
    --cluster-identifier mycluster \
    --preferred-maintenance-window Sun:23:15-Mon:03:15
```

Output:

```
{
    "Cluster": {
        "ClusterIdentifier": "mycluster",
        "NodeType": "dc2.large",
        "ClusterStatus": "available",
        "ClusterAvailabilityStatus": "Available",
        "MasterUsername": "adminuser",
        "DBName": "dev",
        "Endpoint": {
            "Address": "mycluster.cmeaswqeuae.us-west-2.redshift.amazonaws.com",
            "Port": 5439
        },
        "ClusterCreateTime": "2019-12-02T22:45:07.890Z",
        "AutomatedSnapshotRetentionPeriod": 1,
        "ManualSnapshotRetentionPeriod": -1,
        "ClusterSecurityGroups": [
            {
                "ClusterSecurityGroupName": "default",
                "Status": "active"
            }
        ],
        "VpcSecurityGroups": [],
        "ClusterParameterGroups": [
            {
                "ParameterGroupName": "default.redshift-1.0",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "AvailabilityZone": "us-west-2a",
        "PreferredMaintenanceWindow": "sun:23:15-mon:03:15",
        "PendingModifiedValues": {},
        "ClusterVersion": "1.0",
        "AllowVersionUpgrade": true,
        "NumberOfNodes": 2,
        "PubliclyAccessible": true,
        "Encrypted": false,
        "Tags": [],
        "EnhancedVpcRouting": false,
        "IamRoles": [],
        "MaintenanceTrackName": "current"
    }
}
```

For more information, see Maintenance windows in the *Amazon Redshift Management Guide*.

Example 3: To change the master password for the cluster

The following `modify-cluster` example changes the master password for a cluster.

```
aws redshift modify-cluster \
    --cluster-identifier mycluster \
    --master-user-password A1b2c3d4
```

Output:

```
{
    "Cluster": {
        "ClusterIdentifier": "mycluster",
        "NodeType": "dc2.large",
        "ClusterStatus": "available",
        "ClusterAvailabilityStatus": "Available",
        "MasterUsername": "adminuser",
        "DBName": "dev",
        "Endpoint": {
            "Address": "mycluster.cmeaswqeuae.us-west-2.redshift.amazonaws.com",
            "Port": 5439
        },
        "ClusterCreateTime": "2019-12-02T22:45:07.890Z",
        "AutomatedSnapshotRetentionPeriod": 1,
        "ManualSnapshotRetentionPeriod": -1,
        "ClusterSecurityGroups": [
            {
                "ClusterSecurityGroupName": "default",
                "Status": "active"
            }
        ],
        "VpcSecurityGroups": [],
        "ClusterParameterGroups": [
            {
                "ParameterGroupName": "default.redshift-1.0",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "AvailabilityZone": "us-west-2a",
        "PreferredMaintenanceWindow": "sat:03:30-sat:04:00",
        "PendingModifiedValues": {
```

```
            "MasterUserPassword": "****"
        },
        "ClusterVersion": "1.0",
        "AllowVersionUpgrade": true,
        "NumberOfNodes": 2,
        "PubliclyAccessible": true,
        "Encrypted": false,
        "Tags": [],
        "EnhancedVpcRouting": false,
        "IamRoles": [],
        "MaintenanceTrackName": "current"
    }
}
```

For more information, see Modifying a cluster in the *Amazon Redshift Management Guide*.

- For API details, see ModifyCluster in *AWS CLI Command Reference*.

Go

SDK for Go V2

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)
```

```
// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// ModifyCluster sets the preferred maintenance window for the given cluster.
func (actor RedshiftActions) ModifyCluster(ctx context.Context, clusterId string,
 maintenanceWindow string) *redshift.ModifyClusterOutput {
 // Modify the cluster's maintenance window
 input := &redshift.ModifyClusterInput{
  ClusterIdentifier:          aws.String(clusterId),
  PreferredMaintenanceWindow: aws.String(maintenanceWindow),
 }

 var opErr *types.InvalidClusterStateFault
 output, err := actor.RedshiftClient.ModifyCluster(ctx, input)
 if err != nil && errors.As(err, &opErr) {
  log.Println("Cluster is in an invalid state.")
  panic(err)
 } else if err != nil {
  log.Printf("Failed to modify Redshift cluster: %v\n", err)
  panic(err)
 }

 log.Printf("The cluster was successfully modified and now has %s as the
 maintenance window\n", *output.Cluster.PreferredMaintenanceWindow)
 return output
}
```

- For API details, see ModifyCluster in *AWS SDK for Go API Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Modify a cluster.

```
   /**
    * Modifies an Amazon Redshift cluster asynchronously.
    *
    * @param clusterId the identifier of the cluster to be modified
    * @return a {@link CompletableFuture} that completes when the cluster
modification is complete
    */
   public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
clusterId) {
       ModifyClusterRequest modifyClusterRequest =
ModifyClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .preferredMaintenanceWindow("wed:07:30-wed:08:00")
           .build();

       return getAsyncClient().modifyCluster(modifyClusterRequest)
           .whenComplete((clusterResponse, exception) -> {
               if (exception != null) {
                   if (exception.getCause() instanceof RedshiftException) {
                       logger.info("Error: {} ", exception.getMessage());
                   } else {
                       logger.info("Unexpected error: {} ",
exception.getMessage());
                   }
               } else {
                   logger.info("The modified cluster was successfully modified
and has "
                       + clusterResponse.cluster().preferredMaintenanceWindow()
+ " as the maintenance window");
               }
           });
```

```
    }
```

- For API details, see ModifyCluster in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create the client.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Modify a cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { ModifyClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

// Set the parameters
const params = {
  ClusterIdentifier: "CLUSTER_NAME",
  MasterUserPassword: "NEW_MASTER_USER_PASSWORD",
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new ModifyClusterCommand(params));
    console.log("Success was modified.", data);
```

```
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see ModifyCluster in *AWS SDK for JavaScript API Reference*.

Kotlin

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Modify a cluster.

```
suspend fun modifyCluster(clusterId: String?) {
    val modifyClusterRequest =
        ModifyClusterRequest {
            clusterIdentifier = clusterId
            preferredMaintenanceWindow = "wed:07:30-wed:08:00"
        }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.modifyCluster(modifyClusterRequest)
        println(
            "The modified cluster was successfully modified and has
 ${clusterResponse.cluster?.preferredMaintenanceWindow} as the maintenance
 window",
        )
    }
}
```

- For API details, see ModifyCluster in *AWS SDK for Kotlin API reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def modify_cluster(self, cluster_identifier, preferred_maintenance_window):
        """
        Modifies a cluster.

        :param cluster_identifier: The cluster identifier.
        :param preferred_maintenance_window: The preferred maintenance window.
        """
        try:
            self.client.modify_cluster(
                ClusterIdentifier=cluster_identifier,
                PreferredMaintenanceWindow=preferred_maintenance_window,
            )
        except ClientError as err:
            logging.error(
                "Couldn't modify a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

The following code instantiates the RedshiftWrapper object.

```
client = boto3.client("redshift")
redhift_wrapper = RedshiftWrapper(client)
```

- For API details, see ModifyCluster in *AWS SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Modify a cluster.

```
TRY.
    " Example values: iv_cluster_identifier = 'my-redshift-cluster'
    " Example values: iv_pref_maintenance_wn = 'wed:07:30-wed:08:00'
    lo_rsh->modifycluster(
      iv_clusteridentifier = iv_cluster_identifier
      iv_preferredmaintenancewin00 = iv_pref_maintenance_wn
    ).
    MESSAGE 'Redshift cluster modified successfully.' TYPE 'I'.
  CATCH /aws1/cx_rshclustnotfoundfault.
    MESSAGE 'Cluster not found.' TYPE 'I'.
  CATCH /aws1/cx_rshinvcluststatefault.
    MESSAGE 'Invalid cluster state for modification.' TYPE 'I'.
ENDTRY.
```

- For API details, see ModifyCluster in *AWS SDK for SAP ABAP API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Scenarios for Amazon Redshift using AWS SDKs

The following code examples show you how to implement common scenarios in Amazon Redshift with AWS SDKs. These scenarios show you how to accomplish specific tasks by calling multiple functions within Amazon Redshift or combined with other AWS services. Each scenario includes a link to the complete source code, where you can find instructions on how to set up and run the code.

Scenarios target an intermediate level of experience to help you understand service actions in context.

Examples

- Create an Amazon Redshift item tracker
- Get started with serverless data warehouses using the CLI
- Getting started with provisioned data warehouse clusters

Create an Amazon Redshift item tracker

The following code examples show how to create a web application that tracks and reports on work items using an Amazon Redshift database.

Java

SDK for Java 2.x

Shows how to create a web application that tracks and reports on work items stored in an Amazon Redshift database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Redshift data and for use by a React application, see the full example on GitHub.

Services used in this example

- Amazon Redshift

- Amazon SES

Kotlin

SDK for Kotlin

Shows how to create a web application that tracks and reports on work items stored in an Amazon Redshift database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Redshift data and for use by a React application, see the full example on GitHub.

Services used in this example

- Amazon Redshift
- Amazon SES

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Get started with serverless data warehouses using the CLI

The following code example shows how to:

- Use secrets-manager CreateSecret
- Use secrets-manager DeleteSecret
- Use secrets-manager GetSecretValue
- Use redshift CreateNamespace
- Use redshift CreateWorkgroup
- Use redshift DeleteNamespace
- Use iam CreateRole

Bash

AWS CLI with Bash script

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Sample developer tutorials repository.

```
#!/bin/bash

# Amazon Redshift Serverless Tutorial Script with Secrets Manager (No jq
 dependency)
# This script creates a Redshift Serverless environment, loads sample data, and
 runs queries
# Uses AWS Secrets Manager for secure password management without requiring jq

# Set up logging
LOG_FILE="redshift-serverless-tutorial-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift Serverless tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Command failed: $cmd"
    echo "Output: $output"
    cleanup_resources
    exit 1
  fi
}

# Function to generate a secure password that meets Redshift requirements
generate_secure_password() {
  # Redshift password requirements:
  # - 8-64 characters
```

```
  # - At least one uppercase letter
  # - At least one lowercase letter
  # - At least one decimal digit
  # - Can contain printable ASCII characters except /, ", ', \, @, space

  local password=""
  local valid=false
  local attempts=0
  local max_attempts=10

  while [[ "$valid" == false && $attempts -lt $max_attempts ]]; do
    # Generate base password with safe characters
    local base=$(openssl rand -base64 12 | tr -d '/+=' | head -c 12)

    # Ensure we have at least one of each required character type
    local upper=$(echo "ABCDEFGHIJKLMNOPQRSTUVWXYZ" | fold -w1 | shuf -n1)
    local lower=$(echo "abcdefghijklmnopqrstuvwxyz" | fold -w1 | shuf -n1)
    local digit=$(echo "0123456789" | fold -w1 | shuf -n1)
    local special=$(echo "!#$%&*()_+-=[]{}|;:,.<>?" | fold -w1 | shuf -n1)

    # Combine and shuffle
    password="${base}${upper}${lower}${digit}${special}"
    password=$(echo "$password" | fold -w1 | shuf | tr -d '\n')

    # Validate password meets requirements
    if [[ ${#password} -ge 8 && ${#password} -le 64 ]] && \
       [[ "$password" =~ [A-Z] ]] && \
       [[ "$password" =~ [a-z] ]] && \
       [[ "$password" =~ [0-9] ]] && \
       [[ ! "$password" =~ [/\"\'\\@[:space:]] ]]; then
      valid=true
    fi

    ((attempts++))
  done

  if [[ "$valid" == false ]]; then
    echo "ERROR: Failed to generate valid password after $max_attempts attempts"
    exit 1
  fi

  echo "$password"
}
```

```
# Function to create secret in AWS Secrets Manager
create_secret() {
  local secret_name=$1
  local username=$2
  local password=$3
  local description=$4

  echo "Creating secret in AWS Secrets Manager: $secret_name"

  # Create the secret using AWS CLI without jq
  local secret_output=$(aws secretsmanager create-secret \
    --name "$secret_name" \
    --description "$description" \
    --secret-string "{\"username\":\"$username\",\"password\":\"$password\"}" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
 2>&1)

  if echo "$secret_output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Failed to create secret: $secret_output"
    return 1
  fi

  echo "Secret created successfully: $secret_name"
  return 0
}

# Function to retrieve password from AWS Secrets Manager
get_password_from_secret() {
  local secret_name=$1

  # Get the secret value and extract password using sed/grep instead of jq
  local secret_value=$(aws secretsmanager get-secret-value \
    --secret-id "$secret_name" \
    --query 'SecretString' \
    --output text 2>/dev/null)

  if [[ $? -eq 0 ]]; then
    # Extract password from JSON using sed
    echo "$secret_value" | sed -n 's/.*"password":"\([^"]*\)".*/\1/p'
  else
    echo ""
  fi
}
```

```
# Function to wait for a resource to be available
wait_for_resource() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be available..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>/dev/null)
    local status=$(echo "$output" | grep -o '"Status": "[^"]*' | cut -d'"' -f4 ||
 echo "")

    if [[ "$status" == "AVAILABLE" ]]; then
      echo "$resource_type $resource_name is now available"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name status:
 $status. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be
 available"
  return 1
}

# Function to wait for a resource to be deleted
wait_for_resource_deletion() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be deleted..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>&1)

    if echo "$output" | grep -i "not found\|does not exist" > /dev/null; then
```

```
      echo "$resource_type $resource_name has been deleted"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name is still being
 deleted. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be deleted"
  return 1
}

# Function to clean up resources
cleanup_resources() {
  echo ""
  echo "========================================="
  echo "CLEANUP CONFIRMATION"
  echo "========================================="
  echo "The following resources were created:"
  echo "- Redshift Serverless Workgroup: $WORKGROUP_NAME"
  echo "- Redshift Serverless Namespace: $NAMESPACE_NAME"
  echo "- IAM Role: $ROLE_NAME"
  echo "- Secrets Manager Secret: $SECRET_NAME"
  echo ""
  echo "Do you want to clean up all created resources? (y/n): "
  read -r CLEANUP_CHOICE

  if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the workgroup
    echo "Deleting Redshift Serverless workgroup $WORKGROUP_NAME..."
    WORKGROUP_DELETE_OUTPUT=$(aws redshift-serverless delete-workgroup --
workgroup-name "$WORKGROUP_NAME" 2>&1)
    echo "$WORKGROUP_DELETE_OUTPUT"

    # Wait for workgroup to be deleted before deleting namespace
    wait_for_resource_deletion "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-
serverless get-workgroup --workgroup-name $WORKGROUP_NAME"

    # Delete the namespace
    echo "Deleting Redshift Serverless namespace $NAMESPACE_NAME..."
```

```
    NAMESPACE_DELETE_OUTPUT=$(aws redshift-serverless delete-namespace --
namespace-name "$NAMESPACE_NAME" 2>&1)
    echo "$NAMESPACE_DELETE_OUTPUT"

    # Wait for namespace to be deleted
    wait_for_resource_deletion "namespace" "$NAMESPACE_NAME" 20 30 "aws redshift-
serverless get-namespace --namespace-name $NAMESPACE_NAME"

    # Delete the IAM role policy
    echo "Deleting IAM role policy..."
    POLICY_DELETE_OUTPUT=$(aws iam delete-role-policy --role-name "$ROLE_NAME" --
policy-name S3Access 2>&1)
    echo "$POLICY_DELETE_OUTPUT"

    # Delete the IAM role
    echo "Deleting IAM role $ROLE_NAME..."
    ROLE_DELETE_OUTPUT=$(aws iam delete-role --role-name "$ROLE_NAME" 2>&1)
    echo "$ROLE_DELETE_OUTPUT"

    # Delete the secret
    echo "Deleting Secrets Manager secret $SECRET_NAME..."
    SECRET_DELETE_OUTPUT=$(aws secretsmanager delete-secret --secret-id
 "$SECRET_NAME" --force-delete-without-recovery 2>&1)
    echo "$SECRET_DELETE_OUTPUT"

    echo "Cleanup completed."
  else
    echo "Cleanup skipped. Resources will remain in your AWS account."
  fi
}

# Check if required tools are available
if ! command -v openssl &> /dev/null; then
    echo "ERROR: openssl is required but not installed. Please install openssl to
 continue."
    exit 1
fi

# Generate unique names for resources
RANDOM_SUFFIX=$(cat /dev/urandom | tr -dc 'a-z0-9' | head -c 6)
NAMESPACE_NAME="rs-namespace-${RANDOM_SUFFIX}"
WORKGROUP_NAME="rs-workgroup-${RANDOM_SUFFIX}"
ROLE_NAME="RedshiftServerlessS3Role-${RANDOM_SUFFIX}"
SECRET_NAME="redshift-serverless-admin-${RANDOM_SUFFIX}"
```

```
DB_NAME="dev"
ADMIN_USERNAME="admin"

# Generate secure password
echo "Generating secure password..."
ADMIN_PASSWORD=$(generate_secure_password)

# Create secret in AWS Secrets Manager
create_secret "$SECRET_NAME" "$ADMIN_USERNAME" "$ADMIN_PASSWORD" "Admin
 credentials for Redshift Serverless namespace $NAMESPACE_NAME"
if [[ $? -ne 0 ]]; then
  echo "ERROR: Failed to create secret in AWS Secrets Manager"
  exit 1
fi

# Track created resources
CREATED_RESOURCES=()

echo "Using the following resource names:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- IAM Role: $ROLE_NAME"
echo "- Secret: $SECRET_NAME"
echo "- Database: $DB_NAME"
echo "- Admin Username: $ADMIN_USERNAME"
echo "- Admin Password: [STORED IN SECRETS MANAGER]"

# Step 1: Create IAM role for S3 access
echo "Creating IAM role for Redshift Serverless S3 access..."

# Create trust policy document
cat > redshift-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

```
EOF

# Create S3 access policy document
cat > redshift-s3-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role $ROLE_NAME..."
ROLE_OUTPUT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://redshift-trust-policy.json 2>&1)
echo "$ROLE_OUTPUT"
check_error "$ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
CREATED_RESOURCES+=("IAM Role: $ROLE_NAME")

# Attach S3 policy to the role
echo "Attaching S3 access policy to role $ROLE_NAME..."
POLICY_OUTPUT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 S3Access --policy-document file://redshift-s3-policy.json 2>&1)
echo "$POLICY_OUTPUT"
check_error "$POLICY_OUTPUT" "aws iam put-role-policy"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"
```

```
# Step 2: Create a namespace
echo "Creating Redshift Serverless namespace $NAMESPACE_NAME..."
NAMESPACE_OUTPUT=$(aws redshift-serverless create-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --admin-username "$ADMIN_USERNAME" \
  --admin-user-password "$ADMIN_PASSWORD" \
  --db-name "$DB_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$NAMESPACE_OUTPUT"
check_error "$NAMESPACE_OUTPUT" "aws redshift-serverless create-namespace"
CREATED_RESOURCES+=("Redshift Serverless Namespace: $NAMESPACE_NAME")

# Wait for namespace to be available
wait_for_resource "namespace" "$NAMESPACE_NAME" 10 30 "aws redshift-serverless
 get-namespace --namespace-name $NAMESPACE_NAME"

# Associate IAM role with namespace
echo "Associating IAM role with namespace..."
UPDATE_NAMESPACE_OUTPUT=$(aws redshift-serverless update-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --iam-roles "$ROLE_ARN" 2>&1)
echo "$UPDATE_NAMESPACE_OUTPUT"
check_error "$UPDATE_NAMESPACE_OUTPUT" "aws redshift-serverless update-namespace"

# Step 3: Create a workgroup
echo "Creating Redshift Serverless workgroup $WORKGROUP_NAME..."
WORKGROUP_OUTPUT=$(aws redshift-serverless create-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --namespace-name "$NAMESPACE_NAME" \
  --base-capacity 8 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$WORKGROUP_OUTPUT"
check_error "$WORKGROUP_OUTPUT" "aws redshift-serverless create-workgroup"
CREATED_RESOURCES+=("Redshift Serverless Workgroup: $WORKGROUP_NAME")

# Wait for workgroup to be available
wait_for_resource "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-serverless
 get-workgroup --workgroup-name $WORKGROUP_NAME"

# Get workgroup endpoint
WORKGROUP_ENDPOINT=$(aws redshift-serverless get-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --query 'workgroup.endpoint.address' \
  --output text)
```

```
echo "Workgroup endpoint: $WORKGROUP_ENDPOINT"

# Wait additional time for the endpoint to be fully operational
echo "Waiting for endpoint to be fully operational..."
sleep 60

# Step 4: Create tables for sample data
echo "Creating tables for sample data..."

# Create users table
echo "Creating users table..."
USERS_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE users(
    userid INTEGER NOT NULL DISTKEY SORTKEY,
    username CHAR(8),
    firstname VARCHAR(30),
    lastname VARCHAR(30),
    city VARCHAR(30),
    state CHAR(2),
    email VARCHAR(100),
    phone CHAR(14),
    likesports BOOLEAN,
    liketheatre BOOLEAN,
    likeconcerts BOOLEAN,
    likejazz BOOLEAN,
    likeclassical BOOLEAN,
    likeopera BOOLEAN,
    likerock BOOLEAN,
    likevegas BOOLEAN,
    likebroadway BOOLEAN,
    likemusicals BOOLEAN
  );" 2>&1)
echo "$USERS_TABLE_OUTPUT"
check_error "$USERS_TABLE_OUTPUT" "aws redshift-data execute-statement (users
 table)"
USERS_QUERY_ID=$(echo "$USERS_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for query to complete
echo "Waiting for users table creation to complete..."
sleep 5
```

```
# Create event table
echo "Creating event table..."
EVENT_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE event(
    eventid INTEGER NOT NULL DISTKEY,
    venueid SMALLINT NOT NULL,
    catid SMALLINT NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    eventname VARCHAR(200),
    starttime TIMESTAMP
  );" 2>&1)
echo "$EVENT_TABLE_OUTPUT"
check_error "$EVENT_TABLE_OUTPUT" "aws redshift-data execute-statement (event
 table)"
EVENT_QUERY_ID=$(echo "$EVENT_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for query to complete
echo "Waiting for event table creation to complete..."
sleep 5

# Create sales table
echo "Creating sales table..."
SALES_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE sales(
    salesid INTEGER NOT NULL,
    listid INTEGER NOT NULL DISTKEY,
    sellerid INTEGER NOT NULL,
    buyerid INTEGER NOT NULL,
    eventid INTEGER NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    qtysold SMALLINT NOT NULL,
    pricepaid DECIMAL(8,2),
    commission DECIMAL(8,2),
    saletime TIMESTAMP
  );" 2>&1)
echo "$SALES_TABLE_OUTPUT"
check_error "$SALES_TABLE_OUTPUT" "aws redshift-data execute-statement (sales
 table)"
```

```
SALES_QUERY_ID=$(echo "$SALES_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for tables to be created
echo "Waiting for tables to be created..."
sleep 10

# Step 5: Load sample data from Amazon S3
echo "Loading sample data from Amazon S3..."

# Load data into users table
echo "Loading data into users table..."
USERS_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY users
    FROM 's3://redshift-downloads/tickit/allusers_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$USERS_LOAD_OUTPUT"
check_error "$USERS_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 users)"
USERS_LOAD_QUERY_ID=$(echo "$USERS_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for users data loading to complete..."
sleep 10

# Load data into event table
echo "Loading data into event table..."
EVENT_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY event
    FROM 's3://redshift-downloads/tickit/allevents_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$EVENT_LOAD_OUTPUT"
```

```
check_error "$EVENT_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 event)"
EVENT_LOAD_QUERY_ID=$(echo "$EVENT_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for event data loading to complete..."
sleep 10

# Load data into sales table
echo "Loading data into sales table..."
SALES_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY sales
    FROM 's3://redshift-downloads/tickit/sales_tab.txt'
    DELIMITER '\t'
    TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$SALES_LOAD_OUTPUT"
check_error "$SALES_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 sales)"
SALES_LOAD_QUERY_ID=$(echo "$SALES_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for sales data loading to complete..."
sleep 30

# Step 6: Run sample queries
echo "Running sample queries..."

# Query 1: Find top 10 buyers by quantity
echo "Running query: Find top 10 buyers by quantity..."
QUERY1_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT firstname, lastname, total_quantity
    FROM (SELECT buyerid, sum(qtysold) total_quantity
          FROM sales
          GROUP BY buyerid
          ORDER BY total_quantity desc limit 10) Q, users
    WHERE Q.buyerid = userid
```

```
    ORDER BY Q.total_quantity desc;" 2>&1)
echo "$QUERY1_OUTPUT"
check_error "$QUERY1_OUTPUT" "aws redshift-data execute-statement (query 1)"
QUERY1_ID=$(echo "$QUERY1_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 1 to complete..."
sleep 10

# Get query 1 results
echo "Getting results for query 1..."
QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
echo "$QUERY1_STATUS_OUTPUT"
check_error "$QUERY1_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 1)"

QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY1_STATUS" == "FINISHED" ]; then
  QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
  echo "Query 1 Results:"
  echo "$QUERY1_RESULTS"
else
  echo "Query 1 is not yet complete. Status: $QUERY1_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
  QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY1_STATUS" == "FINISHED" ]; then
    QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID"
 2>&1)
    echo "Query 1 Results:"
    echo "$QUERY1_RESULTS"
  else
    echo "Query 1 is still not complete. Status: $QUERY1_STATUS"
  fi
fi
```

```
# Query 2: Find events in the 99.9 percentile in terms of all time total sales
echo "Running query: Find events in the 99.9 percentile in terms of all time
 total sales..."
QUERY2_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT eventname, total_price
    FROM (SELECT eventid, total_price, ntile(1000) over(order by total_price
 desc) as percentile
          FROM (SELECT eventid, sum(pricepaid) total_price
                FROM sales
                GROUP BY eventid)) Q, event E
    WHERE Q.eventid = E.eventid
    AND percentile = 1
    ORDER BY total_price desc;" 2>&1)
echo "$QUERY2_OUTPUT"
check_error "$QUERY2_OUTPUT" "aws redshift-data execute-statement (query 2)"
QUERY2_ID=$(echo "$QUERY2_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 2 to complete..."
sleep 10

# Get query 2 results
echo "Getting results for query 2..."
QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
echo "$QUERY2_STATUS_OUTPUT"
check_error "$QUERY2_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 2)"

QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY2_STATUS" == "FINISHED" ]; then
  QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
  echo "Query 2 Results:"
  echo "$QUERY2_RESULTS"
else
  echo "Query 2 is not yet complete. Status: $QUERY2_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
```

```
  QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
  QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY2_STATUS" == "FINISHED" ]; then
    QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID"
 2>&1)
    echo "Query 2 Results:"
    echo "$QUERY2_RESULTS"
  else
    echo "Query 2 is still not complete. Status: $QUERY2_STATUS"
  fi
fi

# Summary
echo ""
echo "============================================="
echo "TUTORIAL SUMMARY"
echo "============================================="
echo "You have successfully:"
echo "1. Created a Redshift Serverless namespace and workgroup"
echo "2. Created an IAM role with S3 access permissions"
echo "3. Stored admin credentials securely in AWS Secrets Manager"
echo "4. Created tables for sample data"
echo "5. Loaded sample data from Amazon S3"
echo "6. Run sample queries on the data"
echo ""
echo "Redshift Serverless Resources:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- Database: $DB_NAME"
echo "- Endpoint: $WORKGROUP_ENDPOINT"
echo "- Credentials Secret: $SECRET_NAME"
echo ""
echo "To connect to your Redshift Serverless database using SQL tools:"
echo "- Host: $WORKGROUP_ENDPOINT"
echo "- Database: $DB_NAME"
echo "- Username: $ADMIN_USERNAME"
echo "- Password: Retrieve from AWS Secrets Manager secret '$SECRET_NAME'"
echo ""
echo "To retrieve the password from Secrets Manager (without jq):"
echo "aws secretsmanager get-secret-value --secret-id $SECRET_NAME --query
 'SecretString' --output text | sed -n 's/.*\"password\":\"\([^\"]*\)\".*/\1/p'"
```

```
echo ""

# Clean up temporary files
rm -f redshift-trust-policy.json redshift-s3-policy.json

# Clean up resources
cleanup_resources

echo "Tutorial completed at $(date)"
```

- For API details, see the following topics in *AWS CLI Command Reference*.
  - CreateNamespace
  - CreateRole
  - CreateSecret
  - CreateWorkgroup
  - DeleteNamespace
  - DeleteRole
  - DeleteRolePolicy
  - DeleteSecret
  - DeleteWorkgroup
  - GetNamespace
  - GetRole
  - GetSecretValue
  - GetWorkgroup
  - PutRolePolicy
  - UpdateNamespace

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Getting started with provisioned data warehouse clusters

The following code example shows how to:

- Create a Redshift cluster
- Create an IAM role for S3 access
- Create tables and load data
- Run example queries
- Clean up resources

Bash

AWS CLI with Bash script

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Sample developer tutorials repository.

```
#!/bin/bash

# Amazon Redshift Provisioned Cluster Tutorial Script
# This script creates a Redshift cluster, loads sample data, runs queries, and
 cleans up resources
# Version 4: Security improvements and best practices

set -euo pipefail

# Set up logging
LOG_FILE="redshift_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Resources created so far:"
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- Redshift Cluster: $CLUSTER_ID"; fi
    if [ -n "${ROLE_NAME:-}" ]; then echo "- IAM Role: $ROLE_NAME"; fi

    echo "Attempting to clean up resources..."
```

```
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete the cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Deleting Redshift cluster: $CLUSTER_ID"
        aws redshift delete-cluster --cluster-identifier "$CLUSTER_ID" --skip-
final-cluster-snapshot 2>/dev/null || true
        echo "Waiting for cluster deletion to complete..."
        aws redshift wait cluster-deleted --cluster-identifier "$CLUSTER_ID" 2>/
dev/null || true
        echo "Cluster deleted successfully."
    fi

    # Delete the IAM role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Removing IAM role policy..."
        aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 RedshiftS3Access 2>/dev/null || true

        echo "Deleting IAM role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    # Clean up temporary files
    rm -f redshift-trust-policy.json redshift-s3-policy.json

    echo "Cleanup completed."
}

# Trap errors and cleanup
trap 'handle_error "Script interrupted"' INT TERM

# Function to wait for SQL statement to complete
wait_for_statement() {
    local statement_id=$1
    local max_attempts=30
    local attempt=1
    local status=""
```

```
    echo "Waiting for statement $statement_id to complete..."

    while [ $attempt -le $max_attempts ]; do
        status=$(aws redshift-data describe-statement --id "$statement_id" --
query 'Status' --output text 2>/dev/null || echo "")

        if [ "$status" == "FINISHED" ]; then
            echo "Statement completed successfully."
            return 0
        elif [ "$status" == "FAILED" ]; then
            local error=$(aws redshift-data describe-statement --id
 "$statement_id" --query 'Error' --output text 2>/dev/null || echo "Unknown
 error")
            echo "Statement failed with error: $error" >&2
            return 1
        elif [ "$status" == "ABORTED" ]; then
            echo "Statement was aborted." >&2
            return 1
        fi

        echo "Statement status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for statement to complete." >&2
    return 1
}

# Function to check if IAM role is attached to cluster
check_role_attached() {
    local role_arn=$1
    local max_attempts=10
    local attempt=1

    echo "Checking if IAM role is attached to the cluster..."

    while [ $attempt -le $max_attempts ]; do
        local status=$(aws redshift describe-clusters \
            --cluster-identifier "$CLUSTER_ID" \
            --query "Clusters[0].IamRoles[?IamRoleArn=='$role_arn'].ApplyStatus"
 \
```

```
            --output text 2>/dev/null || echo "")

        if [ "$status" == "in-sync" ]; then
            echo "IAM role is successfully attached to the cluster."
            return 0
        fi

        echo "IAM role status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 30
        ((attempt++))
    done

    echo "Timed out waiting for IAM role to be attached." >&2
    return 1
}

# Validate required commands
for cmd in aws jq; do
    if ! command -v "$cmd" &> /dev/null; then
        handle_error "Required command '$cmd' not found. Please install it and
 try again."
    fi
done

# Validate AWS credentials
if ! aws sts get-caller-identity &>/dev/null; then
    handle_error "AWS credentials not configured or invalid"
fi

# Variables to track created resources
CLUSTER_ID="examplecluster"
ROLE_NAME="RedshiftS3Role-$(date +%s)"
DB_NAME="dev"
DB_USER="awsuser"

# Generate secure password using AWS Secrets Manager or random string
if command -v openssl &> /dev/null; then
    DB_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-20)
else
    DB_PASSWORD="TempPass$(date +%s | md5sum | cut -c1-20)"
fi

# Validate password meets requirements
```

```
if [ ${#DB_PASSWORD} -lt 8 ]; then
    handle_error "Generated password does not meet minimum length requirement"
fi

# Store password securely (optional: use AWS Secrets Manager in production)
echo "Generated database password (store securely): $DB_PASSWORD"

echo "=== Step 1: Creating Amazon Redshift Cluster ==="

# Create the Redshift cluster with encryption and audit logging enabled
echo "Creating Redshift cluster: $CLUSTER_ID"
CLUSTER_RESULT=$(aws redshift create-cluster \
  --cluster-identifier "$CLUSTER_ID" \
  --node-type ra3.xlplus \
  --number-of-nodes 2 \
  --master-username "$DB_USER" \
  --master-user-password "$DB_PASSWORD" \
  --db-name "$DB_NAME" \
  --port 5439 \
  --encrypted \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-provisioned \
  2>&1) || handle_error "Failed to create Redshift cluster"

echo "$CLUSTER_RESULT"
echo "Waiting for cluster to become available..."

# Wait for the cluster to be available
aws redshift wait cluster-available --cluster-identifier "$CLUSTER_ID" ||
 handle_error "Timeout waiting for cluster to become available"

# Get cluster status to confirm
CLUSTER_STATUS=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].ClusterStatus' \
  --output text)

echo "Cluster status: $CLUSTER_STATUS"

echo "=== Step 2: Creating IAM Role for S3 Access ==="

# Create trust policy file with restricted permissions
echo "Creating trust policy for Redshift"
cat > redshift-trust-policy.json << 'EOF'
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

chmod 600 redshift-trust-policy.json

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://redshift-trust-policy.json 2>&1) ||
 handle_error "Failed to create IAM role"

echo "$ROLE_RESULT"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Tag the IAM role
echo "Tagging IAM role: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=redshift-provisioned

# Create policy document for S3 access with principle of least privilege
echo "Creating S3 access policy"
cat > redshift-s3-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
```

```
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

chmod 600 redshift-s3-policy.json

# Attach policy to role
echo "Attaching S3 access policy to role"
POLICY_RESULT=$(aws iam put-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-name RedshiftS3Access \
  --policy-document file://redshift-s3-policy.json 2>&1) || handle_error "Failed
 to attach policy to role"

echo "$POLICY_RESULT"

# Attach role to cluster
echo "Attaching IAM role to Redshift cluster"
ATTACH_ROLE_RESULT=$(aws redshift modify-cluster-iam-roles \
  --cluster-identifier "$CLUSTER_ID" \
  --add-iam-roles "$ROLE_ARN" 2>&1) || handle_error "Failed to attach role to
 cluster"

echo "$ATTACH_ROLE_RESULT"

# Wait for the role to be attached
echo "Waiting for IAM role to be attached to the cluster..."
if ! check_role_attached "$ROLE_ARN"; then
    handle_error "Failed to attach IAM role to cluster"
fi

echo "=== Step 3: Getting Cluster Connection Information ==="

# Get cluster endpoint
CLUSTER_INFO=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].Endpoint.{Address:Address,Port:Port}' \
```

```
  --output json)

echo "Cluster endpoint information:"
echo "$CLUSTER_INFO"

echo "=== Step 4: Creating Tables and Loading Data ==="

echo "Creating sales table"
SALES_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS sales; CREATE TABLE sales(salesid integer not null,
 listid integer not null distkey, sellerid integer not null, buyerid integer
 not null, eventid integer not null, dateid smallint not null sortkey, qtysold
 smallint not null, pricepaid decimal(8,2), commission decimal(8,2), saletime
 timestamp);" \
  --query 'Id' --output text)

echo "Sales table creation statement ID: $SALES_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_TABLE_ID"; then
    handle_error "Failed to create sales table"
fi

echo "Creating date table"
DATE_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS date; CREATE TABLE date(dateid smallint not null
 distkey sortkey, caldate date not null, day character(3) not null, week smallint
 not null, month character(5) not null, qtr character(5) not null, year smallint
 not null, holiday boolean default('N'));" \
  --query 'Id' --output text)

echo "Date table creation statement ID: $DATE_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_TABLE_ID"; then
    handle_error "Failed to create date table"
fi
```

```
echo "Loading data into sales table"
SALES_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY sales FROM 's3://redshift-downloads/tickit/sales_tab.txt' DELIMITER
 '\t' TIMEFORMAT 'MM/DD/YYYY HH:MI:SS' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';"
 \
  --query 'Id' --output text)

echo "Sales data load statement ID: $SALES_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_LOAD_ID"; then
    handle_error "Failed to load data into sales table"
fi

echo "Loading data into date table"
DATE_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY date FROM 's3://redshift-downloads/tickit/date2008_pipe.txt'
 DELIMITER '|' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';" \
  --query 'Id' --output text)

echo "Date data load statement ID: $DATE_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_LOAD_ID"; then
    handle_error "Failed to load data into date table"
fi

echo "=== Step 5: Running Example Queries ==="

echo "Running query: Get definition for the sales table"
QUERY1_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT * FROM pg_table_def WHERE tablename = 'sales';" \
  --query 'Id' --output text)

echo "Query 1 statement ID: $QUERY1_ID"
```

```
# Wait for statement to complete
if ! wait_for_statement "$QUERY1_ID"; then
    handle_error "Query 1 failed"
fi

# Get and display results
echo "Query 1 results (first 10 rows):"
aws redshift-data get-statement-result --id "$QUERY1_ID" --max-items 10

echo "Running query: Find total sales on a given calendar date"
QUERY2_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT sum(qtysold) FROM sales, date WHERE sales.dateid = date.dateid
 AND caldate = '2008-01-05';" \
  --query 'Id' --output text)

echo "Query 2 statement ID: $QUERY2_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY2_ID"; then
    handle_error "Query 2 failed"
fi

# Get and display results
echo "Query 2 results:"
aws redshift-data get-statement-result --id "$QUERY2_ID"

echo "=== Tutorial Complete ==="
echo "The following resources were created:"
echo "- Redshift Cluster: $CLUSTER_ID"
echo "- IAM Role: $ROLE_NAME"

echo ""
echo "========================================="
echo "CLEANUP CONFIRMATION"
echo "========================================="
echo "Cleaning up all created resources..."
cleanup_resources
echo "All resources have been cleaned up."

# Securely clear password from memory
```

```
DB_PASSWORD=""

echo "Script completed at $(date)"
```

- For API details, see the following topics in *AWS CLI Command Reference*.
  - CreateCluster
  - CreateRole
  - DeleteCluster
  - DeleteRole
  - DeleteRolePolicy
  - DescribeClusters
  - GetRole
  - ModifyClusterIamRoles
  - PutRolePolicy
  - Wait

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Document history

> **Note**
>
> For a description of new features in Amazon Redshift, see What's new.

The following table describes the important documentation changes to the *Amazon Redshift Management Guide* after June 2018. For notification about updates to this documentation, you can subscribe to an RSS feed.

API version: 2012-12-01

For a list of the changes to the *Amazon Redshift Database Developer Guide*, see Amazon Redshift Database Developer Guide document history.

Change	Description	Date
Amazon Redshift patch 206 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 206.	October 13, 2026
Amazon Redshift patch 205 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 205.	September 16, 2026

Amazon Redshift patch 204 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 204.	August 11, 2026
Amazon Redshift patch 203 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 203.	July 6, 2026
Amazon Redshift patch 202 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 202.	May 28, 2026

Amazon Redshift patch 201 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 201.	April 22, 2026
Amazon Redshift patch 200 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 200.	March 26, 2026
Amazon Redshift patch 199 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 199.	March 4, 2026

Amazon Redshift patch 198 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 198.	February 3, 2026
Update Amazon Redshift managed policies	Added new `AmazonRedshiftFederatedAuthorization` managed policy.	November 21, 2025
Amazon Redshift patch 197 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 197.	November 20, 2025
Amazon Redshift patch 196 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 196.	November 10, 2025

Amazon Redshift patch 195 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 195.	October 24, 2025
Amazon Redshift patch 194 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 194.	October 1, 2025
Amazon Redshift patch 193 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 193.	August 26, 2025

Amazon Redshift patch 192 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 192.	July 30, 2025
Amazon Redshift patch 191 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 191.	June 24, 2025
Amazon Redshift patch 190 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 190.	May 7, 2025
Update Amazon Redshift managed policies	Updates to `AmazonRedshiftServiceLinkedRolePolicy` managed policy with permission `lakeformation:GetDataAccess`.	March 13, 2025

Amazon Redshift patch 189 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 189.	March 7, 2025
Amazon Redshift patch 188 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 188.	February 5, 2025
Amazon Redshift patch 187 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 187.	December 20, 2024
Update Amazon Redshift managed policies	Updates to `AmazonRedshiftServiceLinkedRolePolicy` managed policy with permissions `glue:GetCatalog` and `glue:GetCatalogs`.	December 3, 2024

Amazon Redshift patch 186 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 186.	October 22, 2024
Amazon Redshift patch 185 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 185.	October 9, 2024
Amazon Redshift patch 184 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 184.	September 12, 2024

Amazon Redshift patch 183 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 183.	August 8, 2024
Amazon Redshift patch 182 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 182.	June 26, 2024
Amazon Redshift patch 181 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 181.	May 1, 2024
Update Amazon Redshift managed policies	Update to `AmazonRedshiftServiceLinkedRolePolicy` managed policy with permission `servicequotas:GetServiceQuota` to access AWS quotas or limits.	March 8, 2024

Update query editor v2 managed policies	Updates to `AmazonRedshiftQueryEditorV2FullAccess`, `AmazonRedshiftQueryEditorV2NoSharing`, `AmazonRedshiftQueryEditorV2ReadSharing`, and `AmazonRedshiftQueryEditorV2ReadWriteSharing` managed policies with permissions `redshift-serverless:ListNamespaces` and `redshift-serverless:ListWorkgroups`.	February 21, 2024
Update Amazon Redshift read-only access managed policy	Updates to `AmazonRedshiftReadOnlyAccess` managed policy with permission `redshift:ListRecommendations` to list Amazon Redshift Advisor recommendations.	February 7, 2024
Amazon Redshift patch 180 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 180.	December 29, 2023

Amazon Redshift patch 179 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 179.	November 9, 2023
Update Amazon Redshift managed policies	Updates to `AmazonRedshiftServiceLinkedRolePolicy` managed policy with permissions `ec2:AssignIpv6Addresses` and `ec2:UnassignIpv6Addresses`.	October 31, 2023
Amazon Redshift patch 178 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 178.	September 25, 2023

Update query editor v2 managed policies	Updates to `AmazonRedshiftQueryEditorV2NoSharing`, `AmazonRedshiftQueryEditorV2ReadSharing`, and `AmazonRedshiftQueryEditorV2ReadWriteSharing` managed policies with permissions `sqlworkbench:GetAutocompletionMetadata` and `sqlworkbench:GetAutocompletionResource`.	August 16, 2023
Update Amazon Redshift managed policy	Updates to `AmazonRedshiftServiceLinkedRolePolicy` managed policy to grant permissions on AWS Secrets Manager to create and manage admin credential secrets.	August 14, 2023
Amazon Redshift patch 177 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 177.	August 3, 2023

Amazon Redshift patch 176 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 176.	June 8, 2023
Amazon Redshift patch 175 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 175.	April 28, 2023
Update Amazon Redshift managed policy	Updates to `AmazonRedshiftServiceLinkedRolePolicy` managed policy to remove permissions for ec2 network-related actions. These were specifically associated with the *Purpose:RedshiftMigrateToVpc* resource tag.	April 27, 2023
Update Amazon Redshift Data API managed policy	Updates to `AmazonRedshiftDataFullAccess` managed policy with permission on `redshift:GetClusterCredentialsWithIAM` .	April 7, 2023

Update query editor v2 managed policies	Updates to `AmazonRedshiftQueryEditorV2NoSharing`, `AmazonRedshiftQueryEditorV2ReadSharing`, and `AmazonRedshiftQueryEditorV2ReadWriteSharing` managed policies with permission `sqlworkbench:GetSchemaInference`.	March 21, 2023
Amazon Redshift patch 174 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 174.	March 11, 2023
Update query editor v2 managed policies	Updates to `AmazonRedshiftQueryEditorV2NoSharing`, `AmazonRedshiftQueryEditorV2ReadSharing`, and `AmazonRedshiftQueryEditorV2ReadWriteSharing` managed policies with permission `sqlworkbench:AssociateNotebookWithTab`.	February 2, 2023

Amazon Redshift patch 173 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 173.	January 20, 2023
Amazon Redshift patch 172 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 172.	November 17, 2022
Amazon Redshift patch 171 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 171.	November 9, 2022

Amazon Redshift patch 170 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 170.	July 20, 2022
Amazon Redshift patch 169 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 169.	June 8, 2022
Amazon Redshift patch 168 released.	A new Amazon Redshift patch is being deployed. It takes several weeks for a new version to become available in all Amazon Redshift supported AWS Regions. For more information about this version, see Amazon Redshift patch 168.	April 19, 2022
Support for authentication profiles with Amazon Redshift drivers	You can now connect to Amazon Redshift with an authentication profile.	August 2, 2021

Support for cross-VPC endpoints for Amazon Redshift powered by AWS PrivateLink	You can now use Redshift-managed VPC endpoints with Amazon Redshift.	April 1, 2021
Support for Amazon Redshift query editor enhancements	You can now use the query editor with enhanced VPC routing, longer query run times, and more cluster node types.	February 17, 2021
Support for the console integration with partners	You can integrate with partners using the Amazon Redshift console.	December 9, 2020
Support for the ability to move clusters between Availability Zones	You can now move RA3 clusters between Availability Zones.	December 9, 2020
Support for ra3.xlplus node types	You can now create ra3.xlplus node types.	December 9, 2020
Support for JDBC driver version 2.0	You can now configure the JDBC driver version 2.0.	November 5, 2020
Support for Lambda UDFs and tokenization	You can now can write Lambda UDFs to enable external tokenization of data.	October 26, 2020
Support to schedule the run of an SQL statement	You can now schedule a query on the Amazon Redshift console.	October 22, 2020
Support for the Data API for Amazon Redshift	Amazon Redshift can now be accessed using the built-in Data API. Documentation updates include an *Amazon Redshift Data API Reference*.	September 10, 2020

Support for Amazon Redshift console query monitoring	Updated the guide to describe new query monitoring graphs.	May 7, 2020
Support for usage limits	Updated the guide to describe usage limits.	April 23, 2020
Multi-factor authentication	Updated the guide to describe multi-factor authentication support.	April 20, 2020
Elastic resize now supports node type changes	Updated elastic resize description.	April 6, 2020
Support for ra3.4xlarge node types with managed storage	Updated the guide to include ra3.4xlarge node types.	April 2, 2020
Support for pause and resume	Updated the guide to describe the pause and resume cluster operations.	March 11, 2020
Support for Microsoft Azure AD as an identity provider	Updated the guide to describe the steps to use Microsoft Azure AD as an identity provider.	February 10, 2020
Support for the RA3 node type	Updated the guide to describe the new RA3 node type.	December 3, 2019
Support for the new console	Updated the guide to describe the new Amazon Redshift console.	November 11, 2019
Security information updates	Updates to the security information documentation.	June 24, 2019
Snapshot enhancements	Amazon Redshift now supports several enhancements to managing and scheduling snapshots.	April 4, 2019

Concurrency scaling	You can configure workload management (WLM) to enable concurrency scaling mode. For more information, see Configuring workload management.	March 21, 2019
Updated JDBC and ODBC drivers	Amazon Redshift now supports new versions of the JDBC and ODBC drivers. For more information, see Configure a JDBC connection.	February 4, 2019
Deferred maintenance	If you need to reschedule your cluster's maintenance window, you have the option to defer maintenance by up to 14 days. If we need to update hardware or make other mandatory updates during your period of deferment, we notify you and make the required changes. Your cluster isn't available during these updates. For more information, see Deferring maintenance.	November 20, 2018

Advance notification	Amazon Redshift provides notification in advance for some events. These events have an event category of `pending`. For example, we send an advance notification if a hardware update is required for one of the nodes in your cluster. You can subscribe to pending events the same as other Amazon Redshift events. For more information, see Subscribing to Amazon Redshift event notifications.	November 20, 2018
Elastic resize	Elastic resize is the fastest method to resize a cluster. Elastic resize adds or removes nodes on an existing cluster, then automatically redistributes the data to the new nodes. Because it doesn't create a new cluster, the elastic resize operation completes quickly, usually in a few minutes. For more information, see Resizing clusters.	November 15, 2018
Cancel resize operation	You can now cancel a resize operation while it is in progress. For more information, see Resize operation overview.	November 2, 2018

Modify cluster to change encryption	You can modify an unencrypted cluster to use AWS Key Management Service (AWS KMS) encryption, using either an AWS-managed key or a customer managed key. When you modify your cluster to enable KMS encryption, Amazon Redshift automatically migrates your data to a new encrypted cluster. You can also migrate an unencrypted cluster to an encrypted cluster by modifying the cluster.	October 16, 2018
Amazon Redshift spectrum supports enhanced VPC routing	You can now use Redshift Spectrum with enhanced VPC routing enabled for your cluster. You might need to perform additional configuration steps. For more information, see Using Amazon Redshift spectrum with enhanced VPC routing.	October 10, 2018
Query editor	You can now run SQL queries from the Amazon Redshift Management Console.	October 4, 2018
Workload execution breakdown chart	You can now get a detailed view of your workload's performance by looking at the Workload Execution Breakdown chart in the console. For more information, see Analyzing workload performance.	July 30, 2018

Maintenance tracks	You can now determine if your cluster will always be updated to the latest version of Amazon Redshift or to a previous version by choosing a maintenance track. For more information, see Choosing cluster maintenance tracks.	July 26, 2018

The following table describes the important changes to the *Amazon Redshift Management Guide* before July 2018.

Change	Description	Release date
New CloudWatch metrics	New CloudWatch metrics added for monitoring query performance. For more information, see Performance data in Amazon Redshift	May 17, 2018
HSM encryption	Amazon Redshift supports only AWS CloudHSM for hardware security module (HSM) key management. For more information, see Amazon Redshift database encryption.	March 6, 2018
IAM Role Chaining	If an IAM role attached to your cluster doesn't have access to the necessary resources, you can chain another role, possibly belonging to another account. Your cluster then temporarily assumes the chained role to access the data. You can also grant cross-account access by chaining roles. Each role in the chain assumes the next role in the chain, until the cluster assumes the role at the end of chain. You can chain a maximum of 10 roles. For more information, see Chaining IAM roles in Amazon Redshift.	February 23, 2018
New DC2 node types	The new generation of dense compute (DC) node types offer much better performance at the same	October 17, 2017

Change	Description	Release date
	price as DC1. To take advantage of performance improvements, you can migrate your DC1 cluster to the newer DC2 node types. For more information, see Clusters and nodes in Amazon Redshift.	
ACM certificates	Amazon Redshift is replacing the SSL certificates on your clusters with AWS Certificate Manager (ACM) issued certificates. ACM is a trusted public certifica te authority (CA) that is trusted by most current systems. You might need to update your current trust root CA certificates to continue to connect to your clusters using SSL. For more information, see Transitioning to ACM certificates for SSL connections.	September 18, 2017
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. For more information, see Using service-l inked roles for Amazon Redshift.	September 18, 2017
IAM database user authentication	You can configure your system to permit users to create user credentials and log on to the database based on their IAM credentials. You can also configu re your system to let users sign on using federated single sign-on through a SAML 2.0-compliant identity provider. For more information, see Using IAM authentication to generate database user credentials.	August 11, 2017
Table-level restore supports enhanced VPC routing	Table-level restore is now supported on clusters that use Controlling network traffic with enhanced VPC routing. For more information, see Restoring a table from a snapshot.	July 19, 2017

Change	Description	Release date
Query monitoring rules	Using WLM query monitoring rules, you can define metrics-based performance boundaries for WLM queues and specify what action to take when a query goes beyond those boundaries—log, hop, or abort. You define query monitoring rules as part of your workload management (WLM) configuration. For more information, see Workload management.	April 21, 2017
Enhanced VPC routing	When you use Amazon Redshift enhanced VPC routing, Amazon Redshift forces all COPY and UNLOAD traffic between your cluster and your data repositories through your Amazon VPC. For more information, see Controlling network traffic with Redshift enhanced VPC routing.	September 15, 2016
New connection log fields	The Connection log audit log has two new fields to track SSL connections. If you routinely load audit logs to an Amazon Redshift table, you will need to add the following new columns to the target table: sslcompression and sslexpansion.	May 5, 2016
IAM roles for COPY and UNLOAD	You can now specify one or more AWS Identity and Access Management (IAM) roles that your cluster can use for authentication to access other AWS services. IAM roles provide a more secure alternative to provide authentication with COPY, UNLOAD, or CREATE LIBRARY commands. For more information, see Authorizing Amazon Redshift to access AWS services on your behalf and Authorizing COPY, UNLOAD, CREATE EXTERNAL FUNCTION, and CREATE EXTERNAL SCHEMA operations using IAM roles.	March 29, 2016
Restore from table	You can restore a table from a cluster snapshot to a new table in an active cluster. For more information, see Restoring a table from a snapshot.	March 10, 2016

Change	Description	Release date
Using IAM Condition in policies	You can further restrict access to resources by using the Condition element in IAM policies. For more information, see Using IAM policy conditions for fine-grained access control.	December 10, 2015
Modify publicly accessible	You can modify an existing cluster in a VPC to change whether it is publicly accessible. For more informati on, see Modifying a cluster.	November 20, 2015
Documentation fixes	Published various documentation fixes.	August 28, 2015
Documentation update	Updated troubleshooting guidance about configuri ng network settings to ensure that hosts with different maximum transmission unit (MTU) sizes can determine the packet size for a connection. For more information, see Queries appear to hang and sometimes fail to reach the cluster.	August 25, 2015
Documentation update	Revised entire section about parameter groups for better organization and clarity. For more informati on, see Amazon Redshift parameter groups.	August 17, 2015
WLM dynamic properties	The WLM configuration parameter now supports applying some properties dynamically. Other properties remain static changes and require that associated clusters be rebooted so that the configura tion changes can be applied. For more information, see WLM dynamic and static properties and Amazon Redshift parameter groups.	August 3, 2015
Copy KMS encrypted clusters to another AWS Region	Added content about configuring snapshot copy grants to enable copying of AWS KMS-encrypted clusters to another AWS Region. For more informati on, see Copying AWS KMS–encrypted snapshots to another AWS Region.	July 28, 2015

Change	Description	Release date
Documentation update	Updated the database encryption section to better explain how Amazon Redshift uses AWS KMS or HSMs for managing keys, and how the encryption process works with each of these options. For more information, see Amazon Redshift database encryption.	July 28, 2015
New node type	Amazon Redshift now offers a new node type, DS2. Updated documentation references to existing node types to use new names introduced in this release. Also revised the section to better explain the node type combinations and clarify default quota limits. For more information, see Clusters and nodes in Amazon Redshift.	June 9, 2015
Reserved node offerings	Added content about new reserved node offerings. Also revised the section to better explain and compare the available offerings, and provided examples to demonstrate how on-demand and reserved node pricing affect billing. For more information, see Reserved nodes.	June 9, 2015
Documentation fixes	Published various documentation fixes.	April 30, 2015
New feature	This release of Amazon Redshift introduces new ODBC and JDBC drivers optimized for use with Amazon Redshift. For more information, see Connecting to an Amazon Redshift data warehouse using SQL client tools.	February 26, 2015
New feature	This release of Amazon Redshift introduces cluster performance metrics that allow you to view and analyze query execution details. For more information, see Viewing queries and loads.	February 26, 2015

Change	Description	Release date
Documentation update	Added a new example policy that demonstrates granting permission to common AWS service actions and resources on which Amazon Redshift relies. For more information, see Customer managed policy examples.	January 16, 2015
Documentation update	Updated guidance about setting the maximum transmission unit (MTU) to disable TCP/IP jumbo frames. For more information, see Use EC2 to create your cluster and Queries appear to hang and sometimes fail to reach the cluster.	January 16, 2015
Documentation update	Revised the content about the `wlm_json_configuration` parameter, and provided example syntax to configure this parameter by using the AWS CLI on the Linux, macOS, and Microsoft Windows operating systems. For more information, see Workload management.	January 13, 2015
Documentation update	Added missing event notifications and descripti ons. For more information, see Amazon Redshift provisioned cluster event notifications.	January 8, 2015
Documentation update	Updated guidance about IAM policies for Amazon Redshift actions and resources. Revised the section to improve organization and clarity. For more information, see Security in Amazon Redshift.	November 21, 2014

Change	Description	Release date
New feature	This release of Amazon Redshift introduces the ability to encrypt clusters using encryption keys from AWS Key Management Service (AWS KMS). AWS KMS combines secure, highly available hardware and software to provide a key management system scaled for the cloud. For more information about AWS KMS and encryption options for Amazon Redshift, see Amazon Redshift database encryption and Cluster operations.	November 12, 2014
New feature	This release of Amazon Redshift introduces the ability to tag resources, such as clusters and snapshots. Tags enable you to provide user-defined metadat a to categorize your billing reports based on cost allocation, and to help you better identify resources at a glance. For more information, see Tag resources in Amazon Redshift.	November 4, 2014
New feature	Increased the maximum node limit to 128 nodes for dw1.8xlarge and dw2.8xlarge node sizes. For more information, see Clusters and nodes in Amazon Redshift.	October 30, 2014
New feature	Added the ability to terminate queries and loads from the Amazon Redshift console. For more information, see Viewing queries and loads and Viewing cluster metrics during load operations.	October 28, 2014
Documentation fixes	Published various documentation fixes.	October 17, 2014
New content	Added content about shutting down clusters and deleting clusters. For more information, see Shutting down and deleting a cluster.	August 14, 2014

Change	Description	Release date
Documentation update	Clarified the behavior of the **Allow Version Upgrade** setting for clusters. For more information, see Amazon Redshift provisioned clusters.	August 14, 2014
Documentation update	Revised procedures, screenshots, and organization of topic about working with clusters in Amazon Redshift console. For more information, see Cluster operations.	July 11, 2014
New content	Added a new tutorial about resizing Amazon Redshift clusters, including how to resize a cluster while minimizing the amount of time that the cluster is in read-only mode. For more information, see Resizing a cluster.	June 27, 2014
New feature	Added the ability to rename clusters. For more information, see Renaming a cluster and Modifying a cluster.	June 2, 2014
New feature	Added options to select a different parameter group and security group when you restore a cluster from a snapshot. For more information, see Restoring a cluster from a snapshot.	May 12, 2014
New feature	Added new section to describe how to configure a default Amazon CloudWatch alarm to monitor the percentage of disk space used in an Amazon Redshift cluster. This alarm is a new option in the cluster creation process. For more information, see Default disk space alarm.	April 28, 2014
Documentation update	Clarified information about Elliptic curve Diffie—Hellman Exchange (ECDHE) support in Amazon Redshift. For more information, see SSL.	April 22, 2014

Change	Description	Release date
New feature	Added statement about Amazon Redshift support for the Elliptic curve Diffie—Hellman (ECDH) key agreement protocol. For more information, see SSL.	April 18, 2014
Documentation update	Revised and reorganized the topics in the Connecting to an Amazon Redshift data warehouse using SQL client tools section. Added more information about JDBC and ODBC connections, and a new troubleshooting section for connection issues.	April 15, 2014
Documentation update	Added version in IAM policy examples throughout the guide.	April 3, 2014
Documentation update	Added information about how pricing works when you resize a cluster. For more information, see Reserved nodes.	April 2, 2014
New feature	Added a section about a new parameter, `max_cursor_result_set_size`, which sets the maximum result set size, in megabytes, that can be stored per individual cursor. This parameter value also affects the number of concurrently active cursors for the cluster. For more information, see Amazon Redshift parameter groups.	March 28, 2014
New feature	Added explanation about the **Cluster Version** field now including both cluster engine version and database revision number. For more information, see Amazon Redshift provisioned clusters.	March 21, 2014
New feature	Updated the resize procedure to show the new resize progress information on the cluster's **Status** tab. For more information, see Resizing a cluster.	March 21, 2014

Change	Description	Release date
Documentation update	Reorganized and updated What is Amazon Redshift? and revised Amazon Redshift provisioned clusters overview. Published various documentation fixes.	February 21, 2014
New feature	Added new node types and sizes for Amazon Redshift clusters, and rewrote the related cluster overview topic for better organization and clarity based on feedback. For more information, see Amazon Redshift provisioned clusters.	January 23, 2014
New feature	Added information about using elastic IP (EIP) addresses for publicly-accessible Amazon Redshift clusters in virtual private clouds. For more informati on about EIP in Amazon Redshift, see Redshift resources in a VPC and Creating a Redshift provision ed cluster or Amazon Redshift Serverless workgroup in a VPC.	December 20, 2013
New feature	Added information about the AWS CloudTrail logs for Amazon Redshift. For more information about Amazon Redshift support for CloudTrail, see Logging with CloudTrail.	December 13, 2013
New feature	Added information about the new user activity log and the `enable_user_activity_logging` database parameter for the database audit logging feature in Amazon Redshift. For more information about database audit logging, see Database audit logging. For more information about database pa rameters, see Amazon Redshift parameter groups.	December 6, 2013
New feature	Updated to describe configuring Amazon Redshift to automatically copy automated and manual snapshots to a secondary AWS Region. For more information about configuring cross-Region snapshot copy, see Copying a snapshot to another AWS Region.	November 14, 2013

Change	Description	Release date
New feature	Added section to describe Amazon Redshift audit logging for connection and user activity, and storing these logs in Amazon S3. For more information about database audit logging, see Database audit logging.	November 11, 2013
New feature	Added section to describe Amazon Redshift encryption with new features for managing encryption keys in a hardware security module (HSM) and rotating encryption keys. For more information about encryption, HSM, and key rotation, see Amazon Redshift database encryption, Encryption using hardware security modules, and Encryption key rotation.	November 11, 2013
New feature	Updated to describe publishing notifications of Amazon Redshift events by using Amazon SNS. For information about Amazon Redshift event notifications, see Amazon Redshift provisioned cluster event notifications.	November 11, 2013
New feature	Updated to describe IAM resource level permissions. For information about Amazon Redshift IAM permissions, see Security in Amazon Redshift.	August 9, 2013
New feature	Updated to describe restore progress metrics. For more information, see Restoring a cluster from a snapshot.	August 9, 2013
New feature	Updated to describe cluster snapshot sharing and create snapshot progress metrics. For more information, see Sharing a snapshot.	July 17, 2013
Documentation fixes	Published various documentation fixes.	July 8, 2013

Change	Description	Release date
New console screens	Updated the *Amazon Redshift Management Guide* to match changes in the Amazon Redshift console.	April 22, 2013
New guide	This is the first release of the *Amazon Redshift Management Guide*.	February 14, 2013