Amazon Redshift

Getting Started Guide

Amazon Redshift: Getting Started Guide

Table of Contents

Getting started with Amazon Redshift

Welcome to the *Amazon Redshift Getting Started Guide*. Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the AWS Cloud. 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.

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

- Amazon Redshift management overview – In this topic, you can find an overview of Amazon Redshift.
- Service highlights and pricing – On this product detail page, you can find details about Amazon Redshift service highlights and pricing.
- Amazon Redshift Getting Started Guide (*this guide*) – In this guide, you can find a tutorial of using Amazon Redshift to create a sample cluster and work with sample data.

In this guide, you can find tutorials that walk you through the following:

- Getting started with the Amazon Redshift console (p. 5)
- Connecting to Amazon Redshift (p. 6)
- Getting started with Amazon Redshift clusters and data loading (p. 6)
- Getting started with common database tasks (p. 18)
- Getting started querying your data lake (p. 33)
- Getting started querying data on remote data sources (p. 33)
- Getting started accessing data in other Amazon Redshift clusters (p. 34)
- Getting started training machine learning models with Amazon Redshift data (p. 34)

If your organization is eligible, you might be able to create a cluster under the Amazon Redshift free trial program. To do this, choose **Free trial** to create a configuration with the dc2.large node type. For more information about choosing a free trial, see Amazon Redshift free trial.

Topics

Prerequisites

Before you begin setting up an Amazon Redshift cluster, make sure that you complete the following prerequisites:

Sign up for AWS

If you don't already have an AWS account, sign up for one. If you already have an account, you can skip this prerequisite and use your existing account.

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

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

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

Determine firewall rules

As part of this tutorial, you specify a port when you launch your Amazon Redshift cluster. You also create an inbound ingress rule in a security group to allow access through the port to your cluster.

If your client computer is behind a firewall, make sure that you know an open port that you can use. Using this open port, you can connect to the cluster from a SQL client tool and run queries. If you don't know an open port, work with someone who understands your network firewall rules to determine an open port in your firewall.

Though Amazon Redshift uses port 5439 by default, the connection doesn't work if that port isn't open in your firewall. You can't change the port number for your Amazon Redshift cluster after it's created. Thus, make sure that you specify an open port that works in your environment during the launch process.

This prerequisite applies only when you bring your own data to Amazon Redshift. For more information, see Bringing your own data to Amazon Redshift (p. 10).

Amazon Redshift concepts and data processing flow

In the following sections, you can find key concepts for Amazon Redshift and a description and diagram of the typical Amazon Redshift data processing flow:

Amazon Redshift concepts

Following are some key Amazon Redshift concepts:

- **Cluster** – The core infrastructure component of an Amazon Redshift data warehouse is a cluster.

  A *cluster* is composed of one or more compute nodes. The *compute nodes* run the compiled code.

If a cluster is provisioned with two or more compute nodes, an additional *leader node* coordinates the compute nodes. The leader node handles external communication with applications, such as business intelligence tools and query editors. Your client application interacts directly only with the leader node. The compute nodes are transparent to external applications.

- **Database** – A cluster contains one or more *databases*.

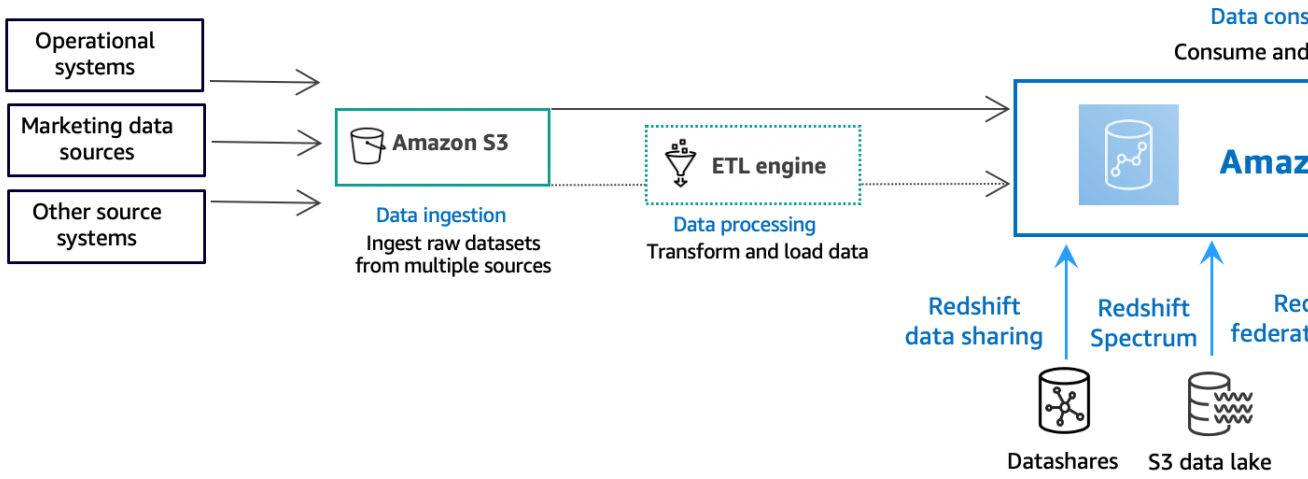
  User data is stored in one or more databases on the compute nodes. Your SQL client communicates with the leader node, which in turn coordinates running queries with the compute nodes. For details about compute nodes and leader nodes, see Data warehouse system architecture. Within a database, user data is organized into one or more schemas.

  Amazon Redshift is a relational database management system (RDBMS) and is compatible with other RDBMS applications. It provides the same functionality as a typical RDBMS, including online transaction processing (OLTP) functions such as inserting and deleting data. Amazon Redshift also is optimized for high-performance batch analysis and reporting of datasets.

Following, you can find a description of typical data processing flow in Amazon Redshift, along with descriptions of different parts of the flow. For further information about Amazon Redshift system architecture, see Data warehouse system architecture.

Typical data processing flow for Amazon Redshift

The following diagram illustrates a typical data processing flow in Amazon Redshift.



An Amazon Redshift *data warehouse* is an enterprise-class relational database query and management system. Amazon Redshift supports client connections with many types of applications, including business intelligence (BI), reporting, data, and analytics tools. When you run analytic queries, you are retrieving, comparing, and evaluating large amounts of data in multiple-stage operations to produce a final result.

At the *data ingestion* layer, different types of data sources continuously upload structured, semistructured, or unstructured data to the data storage layer. This data storage area serves as a staging area that stores data in different states of consumption readiness. An example of storage might be an Amazon Simple Storage Service (Amazon S3) bucket.

At the optional *data processing* layer, the source data goes through preprocessing, validation, and transformation using extract, transform, load (ETL) or extract, load, transform (ELT) pipelines. These raw datasets are then refined by using ETL operations. An example of an ETL engine is AWS Glue.

At the *data consumption* layer, data is loaded into your Amazon Redshift cluster, where you can run analytical workloads.

Data can also be consumed for analytical workloads as follows:

- Use *datashares* to share live data across Amazon Redshift clusters for read purposes with relative security and ease. You can share data at different levels, such as databases, schemas, tables, views (including regular, late-binding, and materialized views), and SQL user-defined functions (UDFs).

  For more information about data sharing, see Getting started accessing data in other Amazon Redshift clusters (p. 34).
- Use Amazon Redshift Spectrum to query data in Amazon S3 files without having to load the data into Amazon Redshift tables. Amazon Redshift provides SQL capability designed for fast online analytical processing (OLAP) of very large datasets that are stored in both Amazon Redshift clusters and Amazon S3 data lakes.

  For more information about Redshift Spectrum, see Getting started querying your data lake (p. 33).
- Join data from relational databases, such as Amazon Relational Database Service (Amazon RDS) and Amazon Aurora, or Amazon S3, with data in your Amazon Redshift database using a federated query. You can use Amazon Redshift to query operational data directly (without moving it), apply transformations, and insert data into your Amazon Redshift tables.

  For more information about federated queries, see Getting started querying data on remote data sources (p. 33).
- Amazon Redshift machine learning (ML) creates models, using data you provided and metadata associated with data inputs. These models capture patterns in the input data. You can use these models to generate predictions for new input data. Amazon Redshift works with Amazon SageMaker Autopilot to automatically get the best model and make the prediction function available in Amazon Redshift.

  For more information about Amazon Redshift ML, see Getting started training machine learning models with Amazon Redshift data (p. 34).

Getting started with Amazon Redshift basics

If you are a first-time user of Amazon Redshift, we recommend that you read the following sections to help you get started using Amazon Redshift.

Topics

Getting started with the Amazon Redshift console

After you have signed in to the Amazon Redshift console, you can create and manage all Amazon Redshift objects, including clusters, databases, and nodes. You can also view queries, run queries, and perform other data definition language (DDL) and data manipulation language (DML) operations.

If you are a first-time user of Amazon Redshift, we recommend that you begin by going to the **Dashboard**, **Clusters**, and **query editor v2** pages to get started using the console.

To get started with the Amazon Redshift console, watch the following video: Getting stated with Amazon Redshift.

Following, you can find a screenshot of the Amazon Redshift console and descriptions of its sections.

Following, you can find descriptions of the navigation pane items of the Amazon Redshift console:

- **Amazon Redshift serverless** – Access and analyze data without the need to set up, tune, and manage Amazon Redshift provisioned clusters.
- **Provisioned clusters dashboard** – Check **Cluster metrics** and **Query overview** for insights to metrics data (such as CPU utilization) and query information. Using these can help you determine if your performance data is abnormal over a specified time range.
- **Clusters** – View a list of clusters in your AWS account, choose a cluster to start querying, or perform cluster-related actions. You can also create a new cluster from this page.
- **Query editor** – Run queries on databases hosted on your Amazon Redshift cluster, save queries for reuse, or schedule them to run at a future time (in the query editor only).
- **Query editor v2** – Use the query editor v2 that is a separate web-based SQL client application to author and run queries on your Amazon Redshift data warehouse. You can visualize your results in charts and collaborate by sharing your queries with others on your team.
- **Queries and loads** – Get information for reference or troubleshooting, such as a list of recent queries and the SQL text for each query.
- **Datashares** – As a producer account administrator, either authorize consumer accounts to access datashares or choose not to authorize access. To use an authorized datashare, a consumer account administrator can associate the datashare with either an entire AWS account or specific cluster namespaces in an account. An administrator can also decline a datashare.
- **Configurations** – Connect to Amazon Redshift clusters from SQL client tools over Java Database Connectivity (JDBC) and Open Database Connectivity (ODBC) connections. You can also set up

an Amazon Redshift–managed virtual private cloud (VPC) endpoint. Doing so provides a private connection between a VPC based on the Amazon VPC service that contains a cluster and another VPC that is running a client tool.

- **Advisor** – Get specific recommendations about changes you can make to your Amazon Redshift cluster to prioritize your optimizations.
- **AWS Marketplace** – Get information on other tools or AWS services that work with Amazon Redshift.
- **Alarms** – Create alarms on cluster metrics to view performance data and track metrics over a time period that you specify.
- **Events** – Track events and get reports on information such as the date the event occurred, a description, or the event source.
- **What's new** – View new Amazon Redshift features and product updates.

Connecting to Amazon Redshift

To connect to Amazon Redshift clusters, from the **Clusters** page, expand **Connect to Amazon Redshift clusters** and do one of the following:

- Use the query editor v2 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 v2.

  For more information, see Querying a database using the Amazon Redshift query editor v2.
- Connect to Amazon Redshift from your client tools using JDBC or ODBC drivers by copying the JDBC or ODBC driver URL.

  To work with data in your cluster, you need JDBC or ODBC drivers for connectivity from your client computer or instance. Code your applications to use JDBC or ODBC data access API operations, or use SQL client tools that support either JDBC or ODBC.

  For more information on how to find your cluster connection string, see Finding your cluster connection string.
- If your SQL client tool requires a driver, you can download an operating system-specific driver to connect to Amazon Redshift from your client tools.

  For more information on how to install the appropriate driver for your SQL client, see Configuring a JDBC driver version 2.0 connection.

  For more information on how to configure an ODBC connection, see Configuring an ODBC connection.

Getting started with Amazon Redshift clusters and data loading

In this section, you can find two tutorials that walk you through the process of creating a sample Amazon Redshift cluster. In one, you use a sample dataset, and in the other you bring your own dataset.

Make sure that you have the prerequisites before getting started. For more information, see Prerequisites (p. 1).

Topics

Using a sample dataset

In this tutorial, you walk through the process to create an Amazon Redshift cluster by using a sample dataset. Amazon Redshift automatically loads the sample dataset when you are creating a new cluster. You can immediately query the data after the cluster is created.

Before you begin setting up an Amazon Redshift cluster, make sure that you complete the Sign up for AWS (p. 2) and Determine firewall rules (p. 2).

In this tutorial, you perform the steps shown following:

Topics

> **Important**
> The sample cluster that you create runs in a live environment. The on-demand rate is $0.25 per hour for using the sample cluster that is designed in this tutorial until you delete it. For more pricing information, see Amazon Redshift pricing. If you have questions or get stuck, you can contact the Amazon Redshift team by posting on our Discussion forum.

This tutorial isn't meant for production environments and doesn't discuss options in depth. After you complete the steps in this tutorial, you can use Additional resources (p. 35) to find more in-depth information. This information can help you plan, deploy, and maintain your clusters, and work with the data in your data warehouse.

Step 1: Create a sample Amazon Redshift cluster

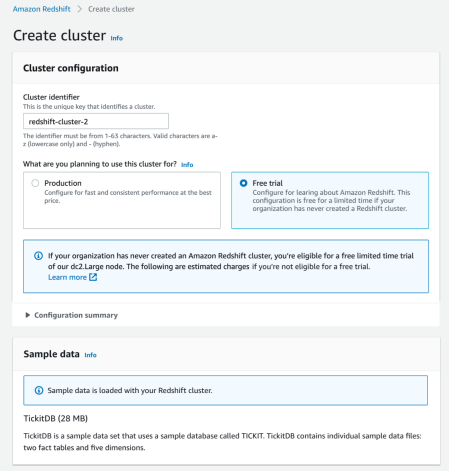
When you have the prerequisites completed, you can start creating your Amazon Redshift cluster, based on a sample dataset.

To create an Amazon Redshift cluster based on a sample dataset:

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshift/.
2. To create a cluster, do one of the following:
   - On the Amazon Redshift service page, choose **Create cluster**. The Create cluster page appears.
   - On the https://console.aws.amazon.com/redshift/, choose **DASHBOARD**, then choose **Create cluster**.
   - On the https://console.aws.amazon.com/redshift/, choose **CLUSTERS**, then choose **Create cluster**.
3. In the **Cluster configuration** section, specify a **Cluster identifier**. This identifier must be unique. The identifier must be from 1–63 characters using as valid characters a–z (lowercase only) and - (hyphen).

   Enter **`examplecluster`** for this tutorial.
4. If your organization is eligible, you might be able to create a cluster under the Amazon Redshift free trial program. To do this, choose **Free trial** to create a configuration with the dc2.large node type. For more information about choosing a free trial, see Amazon Redshift free trial.

   The console displays your selection, as shown in the screenshot following.



If you later choose another node type, your organization is no longer eligible for the free trial.

After you choose your node type, do one of the following:

- 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. You can start using the query editor v2 to query data.
- To bring your own data to your Amazon Redshift cluster, choose **Production**. Then, in **Sample data**, choose **Load sample data**. For information about bringing your own data, see Bringing your own data to Amazon Redshift (p. 10).

Amazon Redshift automatically loads the sample dataset into your sample Amazon Redshift cluster.

5. In the **Database configuration** section, specify values for **Admin user name** and **Admin user password**. Or choose **Generate password** to use a password generated by Amazon Redshift.

   For this tutorial, use these values:

   - **Admin user name**: Enter **`awsuser`**.
   - **Admin user password**: Enter a value for the password.
6. Choose **Create cluster**.

This tutorial uses the Amazon Redshift query editor v2. You can use this editor to query data immediately, after Amazon Redshift finishes creating the cluster.

You can also choose other SQL client tools that support JDBC or ODBC drivers to work with data in your cluster. For more information, see Connecting to an Amazon Redshift cluster using SQL client tools in the *Amazon Redshift Management Guide*.

Step 2: Try example queries using the query editors

When Amazon Redshift is creating your Amazon Redshift cluster, it automatically uploads the sample dataset Tickit. Cluster creation might take a few minutes to complete. After creation completes, the cluster status becomes ACTIVE. You can view the sample Tickit tables from the sample dataset.

Using the query editor

You can view the sample Tickit tables in the query editor v2 by choosing the cluster, the `dev` database, and `public` schema.

After the Amazon Redshift cluster is created, in **Connect to Amazon Redshift clusters**, choose **Query data**.

From the query editor v2, connect to a database, and choose the cluster name in the tree-view panel. If prompted, enter the connection parameters.

When you connect to a cluster and its databases, you provide a **Database** name and **User name**. You also provide parameters required for one of the following authentication methods:

Database user name and password

With this method, also provide a **Password** for the database that you are connecting to.

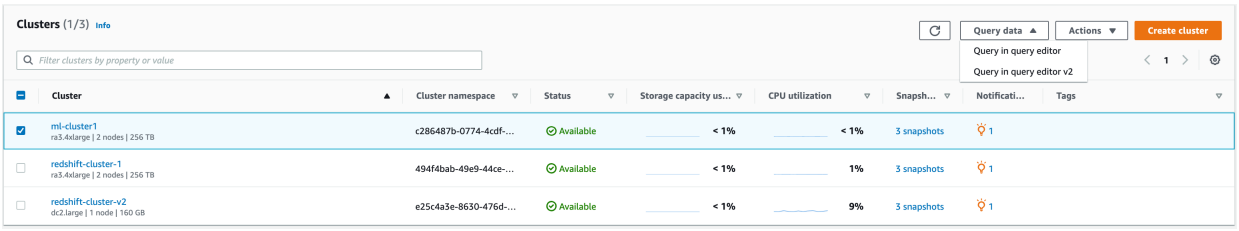
Temporary credentials

With this method, query editor v2, generates a temporary password to connect to the database.

When you select a cluster with query editor v2, depending on the context, you can create, edit, and delete connections using the context (right-click) menu.

By default, Amazon Redshift creates a default database named `dev` and a default schema named `public`. To view the individual data files of the sample dataset, choose a cluster, go to the query editor v2, and choose the `dev` database, `public` schema, then `Tables`.

Alternatively, in the navigation pane, choose **Clusters** and the cluster you want query data on. Then under **Query data**, choose either **Query in query editor** or **Query in query editor v2** to query data in your specified query editor.



Trying example queries

Try some example queries in one of the query editors, as shown following. For more information on working with the SELECT command, see SELECT in the Amazon Redshift Database Developer Guide.

```
-- Find total sales on a given calendar date.
SELECT sum(qtysold)
FROM   sales, date
WHERE  sales.dateid = date.dateid
AND    caldate = '2008-01-05';

-- Find top 10 buyers by quantity.
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;

-- Find events in the 99.9 percentile in terms of all time gross sales.
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;
```

After you complete the steps in this tutorial, you can use Additional resources (p. 35) to find more in-depth information. This information can help you plan, deploy, and maintain your clusters, and work with the data in your data warehouse.

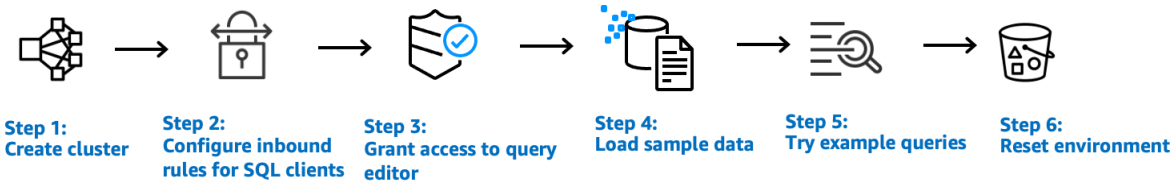
You can also try the Bringing your own data to Amazon Redshift (p. 10) tutorial to create a cluster with your own dataset.

Bringing your own data to Amazon Redshift

In this tutorial, you walk through the process to create an Amazon Redshift cluster by bringing your own dataset to Amazon Redshift. You can use this sample cluster to evaluate the Amazon Redshift service.

Before you begin setting up an Amazon Redshift cluster, make sure that you complete the Sign up for AWS (p. 2) and Determine firewall rules (p. 2).

In this tutorial, you perform the steps shown following.



Important
The sample cluster that you create runs in a live environment. The on-demand rate is $0.25 per hour for using the sample cluster that is designed in this tutorial until you delete it. For more pricing information, go to the Amazon Redshift pricing page. If you have questions or get stuck, you can contact the Amazon Redshift team by posting on our Discussion forum.

This tutorial isn't meant for production environments and doesn't discuss options in depth. After you complete the steps in this tutorial, you can use Additional resources (p. 35) to find more in-depth information. This information can help you plan, deploy, and maintain your clusters, and work with the data in your data warehouse.

Topics

Step 1: Create a sample Amazon Redshift cluster

For any operation that accesses data from another AWS resource, your cluster needs permission to access the resource and the data on the resource on your behalf. An example is using a COPY command to load data from Amazon Simple Storage Service (Amazon S3). You provide those permissions by using AWS Identity and Access Management (IAM). You can do this through an IAM role that is attached to your cluster. Or you can provide the AWS access key for an IAM user that has the necessary permissions. For more information about credentials and access permissions, see Credentials and access permissions.

To best protect your sensitive data and safeguard your AWS access credentials, we recommend creating an IAM role and attaching it to your cluster. For more information about providing access permissions, see Permissions to access other AWS resources.

The cluster that you are about to create is live, not running in a sandbox. You incur the standard Amazon Redshift usage fees for the cluster until you delete it. If you complete the tutorial described here in one sitting and delete the cluster when you are finished, the total charges are minimal.

To create an Amazon Redshift cluster

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

   Important
   If you use IAM user credentials, ensure that you have the necessary permissions to perform the cluster operations. For more information, see Controlling access to IAM users in the *Amazon Redshift Management Guide*.
2. At upper right, choose the AWS Region where you want to create the cluster.
3. On the navigation menu, choose **Clusters**, then choose **Create cluster**. The **Create cluster** page appears.
4. In the **Cluster configuration** section, specify values for **Cluster identifier**, **Node type**, and **Nodes**:

   - **Cluster identifier**: Enter **`examplecluster`** for this tutorial. This identifier must be unique. The identifier must be from 1–63 characters using as valid characters a–z (lowercase only) and - (hyphen).
   - Choose one of the following methods to size your cluster:

     Note
     The following step assumes an AWS Region that supports RA3 node types. For a list of AWS Regions that support RA3 node types, see Overview of RA3 node types in the *Amazon Redshift Management Guide*.

     - If your AWS Region supports RA3 node types, choose either **Production** or **Free trial** to answer the question **What are you planning to use this cluster for?**

       If your organization is eligible, you might be able to create a cluster under the Amazon Redshift free trial program. For information about creating clusters using the free trial program, see Using a sample dataset (p. 7). To do this, choose **Free trial** to create a configuration with the dc2.large node type. For more information about choosing a free trial, see Amazon Redshift free trial.
     - If you don't know how large to size your cluster, choose **Help me choose**. Doing so 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.
   - Choose **dc2.large** for **Node type** and **2** for **Nodes** for this tutorial.
   - If you have chosen **Production** for your cluster, then do one of the following:
     - To use the sample dataset Amazon Redshift provides, in **Sample data**, choose **Load sample data**. Amazon Redshift loads the sample dataset Tickit to the default `dev` database and `public` schema.
     - To bring your own data to Amazon Redshift, continue with the rest of the tutorial.
5. In the **Database configuration** section, specify values for **Database name (optional)**, **Database port (optional)**, **Admin user name**, and **Admin user password**. Or choose **Generate password** to use a password generated by Amazon Redshift.

For this tutorial, use these values:

- **Database name (optional)**: Enter **`dev`**.
- **Database port (optional)**: Enter **`5439`**.
- **Admin user name**: Enter **`awsuser`**.
- **Admin user password**: Enter a value for the password.

6. For this tutorial, create an IAM role and set it as the default for your cluster, as described following. There can only be one default IAM role set for a cluster.

   a. Under **Cluster permissions**, for **Manage IAM roles**, choose **Create IAM role**.

   b. Specify an Amazon S3 bucket for the IAM role to access by one of the following methods:

      - Choose **No additional Amazon S3 bucket** to allow the created IAM role to access only the Amazon S3 buckets that are named as `redshift`.
      - Choose **Any Amazon S3 bucket** to allow the created IAM role to access all Amazon S3 buckets.
      - Choose **Specific Amazon S3 buckets** to specify one or more Amazon S3 buckets for the created IAM role to access. Then choose one or more Amazon S3 buckets from the table.

   c. Choose **Create IAM role as default**. Amazon Redshift automatically creates and sets the IAM role as the default for your cluster.

      Because you created your IAM role from the console, it has the `AmazonRedshiftAllCommandsFullAccess` policy attached. This allows Amazon Redshift to copy, load, query, and analyze data from Amazon resources in your IAM account.

   For information on how to manage the default IAM role for a cluster, see Creating an IAM role as default for Amazon Redshift.

7. (Optional) In the **Additional configurations** section, turn off **Use defaults** to modify **Network and security**, **Database configuration**, **Maintenance**, **Monitoring**, and **Backup** settings.

   In some cases, you might create your cluster with the **Load sample data** option and want to turn on enhanced Amazon VPC routing. If so, the cluster in your virtual private cloud (VPC) requires access to the Amazon S3 endpoint for data to be loaded.

   To make the cluster publicly accessible, you can do one of two things. You can configure a network address translation (NAT) address in your VPC for the cluster to access the internet. Or you can configure an Amazon S3 VPC endpoint in your VPC. For more information about enhanced Amazon VPC routing, see Enabling enhanced Amazon VPC routing in the *Amazon Redshift Management Guide*.

8. Choose **Create cluster**.

Step 2: Configure inbound rules for SQL clients

Later in this tutorial, you access your cluster from within a virtual private cloud (VPC) based on the Amazon VPC service. However, if you use an SQL client from outside your firewall to access the cluster, make sure that you grant inbound access.

You can skip this step if you plan to access the cluster with the Amazon Redshift query editor from within your VPC.

To check your firewall and grant inbound access to your cluster

1. Check your firewall rules if your cluster needs to be accessed from outside a firewall. For example, your client might be an Amazon Elastic Compute Cloud (Amazon EC2) instance or an external computer.

   For more information on firewall rules, see Security group rules in the *Amazon EC2 User Guide for Linux Instances*.
2. To access from an Amazon EC2 external client, add an ingress rule to the security group attached to your cluster that allows inbound traffic. You add Amazon EC2 security group rules in the Amazon EC2 console. For example, a CIDR/IP of 192.0.2.0/24 allows clients in that IP address range to connect to your cluster. Find out the correct CIDR/IP for your environment.

Step 3: Grant access to one of the query editors and run queries

To query databases hosted by your Amazon Redshift cluster, you have two options:

1. Connect to your cluster and run queries on the AWS Management Console with one of the query editors.

   If you use one of the query editors, you don't have to download and set up an SQL client application.
2. Connect to your cluster through an SQL client tool, such as SQL Workbench/J. For more information about using SQL Workbench/J, see Connect to your cluster by using SQL Workbench/J in the *Amazon Redshift Management Guide.*

Using one of the Amazon Redshift query editors is the easiest way to run queries on databases hosted by your Amazon Redshift cluster. After creating your cluster, you can immediately run queries using the Amazon Redshift console. For details about considerations when using the Amazon Redshift query editor, see Querying a database using the query editor in the *Amazon Redshift Management Guide.*

Granting access to the query editor v2

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. Or an administrator can use a customer managed key by choosing the Amazon Resource Name (ARN) for the key in the configuration page. After you configure an account, AWS KMS encryption settings can't be changed. For more information, see Configuring your AWS account.

To access the query editor v2, you need permission. An administrator can attach one of the following AWS-managed policies to the IAM user or role to grant permissions. 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.

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 more information, see Accessing the query editor v2.

Using the query editor v2

To use the query editor v2 to query a database, see Working with query editor v2.

Granting access to the query editor

To use the Amazon Redshift query editor, you need permission. To set access, attach the `AmazonRedshiftQueryEditor` and `AmazonRedshiftReadOnlyAccess` IAM policies to the IAM user that you use to access your cluster.

If you have already created an IAM user to access Amazon Redshift, you can attach the `AmazonRedshiftQueryEditor` and `AmazonRedshiftReadOnlyAccess` policies to that user. If you haven't created an IAM user yet, create one and attach the policies to the IAM user.

To attach the required IAM policies for the query editor

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose **Users**.
3. Choose the user that needs access to the query editor.
4. Choose **Add permissions**.
5. Choose **Attach existing policies directly**.
6. For **Policy names**, choose **AmazonRedshiftQueryEditor** and **AmazonRedshiftReadOnlyAccess**.
7. Choose **Next: Review**.
8. Choose **Add permissions**.

Using the query editor

To use the query editor to query a database, see Querying a database using the query editor.

Step 4: Load data from Amazon S3 to Amazon Redshift

Using one of the Amazon Redshift query editors is the easiest way to load data to tables. After creating your cluster, you can load data from Amazon S3 to your cluster using the Amazon Redshift console.

Using the query editor v2 simplifies loading data when using the Load data wizard. For more information, see Loading your own data from Amazon S3 to Amazon Redshift using the query editor v2 (p. 14). You can also use the query editor v2 to create tables and load your data. For more information, see Loading sample data from Amazon S3 using the query editor (p. 14).

Loading your own data from Amazon S3 to Amazon Redshift using the query editor v2

To load your own data from Amazon S3 to Amazon Redshift, Amazon Redshift requires an IAM role that has the required privileges to load data from the specified Amazon S3 bucket.

First, connect to a database. Next, create some tables in the database. Then load your own data from Amazon S3 to Amazon Redshift. For more information on how to work with the query editor v2, see Working with query editor v2 in the *Amazon Redshift Management Guide*.

The COPY command generated and used in the query editor v2 Load data wizard supports all the parameters available to the COPY command syntax to load data from Amazon S3. For information about the COPY command and its options used to copy load from Amazon S3, see COPY from Amazon Simple Storage Service in the *Amazon Redshift Database Developer Guide*.

Loading sample data from Amazon S3 using the query editor

At this point, you have a database called `dev` and you are connected to it. Next, you create some tables in the database, upload data to the tables, and try a query. For your convenience, the sample data that you load is available in an Amazon S3 bucket.

Note
If you're using a SQL client tool, ensure that your SQL client is connected to the cluster.

After you complete this step, you can do the following:

- Try example queries at Step 5: Try example queries using the query editor (p. 17).
- Reset your environment at Step 6: Reset your environment (p. 18).
- Find more information about Amazon Redshift at Additional resources (p. 35).

Note
To try querying data in the query editor without loading your own data, choose **Load sample data** in **Sample data**. If you do, Amazon Redshift loads its sample dataset to your Amazon Redshift cluster automatically during cluster creation.

To load sample data from Amazon S3

1. Create tables.

   If you are using the Amazon Redshift query editor, individually copy and run the following create table statements to create tables in the `dev` database. For more information about the syntax, see CREATE TABLE in the *Amazon Redshift Database Developer Guide*.

```
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);
```

```
create table venue(
 venueid smallint not null distkey sortkey,
 venuename varchar(100),
 venuecity varchar(30),
 venuestate char(2),
 venueseats integer);
```

```
create table category(
 catid smallint not null distkey sortkey,
 catgroup varchar(10),
 catname varchar(10),
 catdesc varchar(50));
```

```
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'));
```

```
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);
```

```
create table listing(
 listid integer not null distkey,
 sellerid integer not null,
 eventid integer not null,
 dateid smallint not null  sortkey,
 numtickets smallint not null,
 priceperticket decimal(8,2),
 totalprice decimal(8,2),
 listtime timestamp);
```

```
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. Load sample data from Amazon S3 by using the COPY command.

   Note
   We recommend using the COPY command to load large datasets into Amazon Redshift from Amazon S3 or Amazon DynamoDB. For more information about COPY syntax, see COPY in the *Amazon Redshift Database Developer Guide*.

   a. Download the file tickitdb.zip, which contains individual sample data files.

   b. Unzip and load the individual files to a `tickit` folder in your Amazon S3 bucket in your AWS Region.

   c. Edit the COPY commands in this tutorial to point to the files in your Amazon S3 bucket. For information about how to manage files with Amazon S3, see Creating and configuring an S3 Bucket in the *Amazon Simple Storage Service User Guide*.

   d. Provide authentication for your cluster to access Amazon S3 on your behalf to load the sample data. You provide authentication by referencing the IAM role that you created and set as the default for your cluster in previous steps.

      The COPY commands include a placeholder for the Amazon Resource Name (ARN) for the IAM role, your bucket name, and an AWS Region, as shown in the following example.

```
copy users from 's3://<myBucket>/tickit/allusers_pipe.txt'
iam_role default
delimiter '|' region '<aws-region>';
```

Your COPY command should look similar to the following example.

```
copy users from 's3://<myBucket>/tickit/allusers_pipe.txt'
iam_role default
delimiter '|' region '<aws-region>';
```

To load the sample data, replace *<myBucket>* and *<aws-region>* in the following COPY commands with your values. If you are using the Amazon Redshift query editor, individually run the following commands.

```
copy users from 's3://<myBucket>/tickit/allusers_pipe.txt'
iam_role default
delimiter '|' region '<aws-region>';
```

```
copy venue from 's3://<myBucket>/tickit/venue_pipe.txt'
iam_role default
delimiter '|' region '<aws-region>';
```

```
copy category from 's3://<myBucket>/tickit/category_pipe.txt'
iam_role default
delimiter '|' region '<aws-region>';
```

```
copy date from 's3://<myBucket>/tickit/date2008_pipe.txt'
iam_role default
delimiter '|' region '<aws-region>';
```

```
copy event from 's3://<myBucket>/tickit/allevents_pipe.txt'
iam_role default
delimiter '|' timeformat 'YYYY-MM-DD HH:MI:SS' region '<aws-region>';
```

```
copy listing from 's3://<myBucket>/tickit/listings_pipe.txt'
iam_role default
delimiter '|' region '<aws-region>';
```

```
copy sales from 's3://<myBucket>/tickit/sales_tab.txt'
iam_role default
delimiter '\t' timeformat 'MM/DD/YYYY HH:MI:SS' region '<aws-region>';
```

Step 5: Try example queries using the query editor

Now, try some example queries, as shown following. For more information on working with the SELECT command, see SELECT in the *Amazon Redshift Developer Guide*.

```
-- Get definition for the sales table.
SELECT *
FROM pg_table_def
WHERE tablename = 'sales';

-- Find total sales on a given calendar date.
SELECT sum(qtysold)
FROM   sales, date
WHERE  sales.dateid = date.dateid
```

```
AND    caldate = '2008-01-05';

-- Find top 10 buyers by quantity.
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;

-- Find events in the 99.9 percentile in terms of all time gross sales.
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;
```

You have successfully created an Amazon Redshift cluster and queried data from your own dataset using the Amazon Redshift query editor.

Step 6: Reset your environment

When you have completed this tutorial, we suggest that you reset your environment to the previous state by deleting your sample cluster. You continue to incur charges for the Amazon Redshift service until you delete the cluster.

However, you might want to keep the sample cluster running if you intend to try tasks in other Amazon Redshift guides.

To delete a cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshift/.
2. On the navigation menu, choose **Clusters** to display your list of clusters.
3. Choose the **`examplecluster`** cluster. For **Actions**, choose **Delete**. The **Delete cluster** page appears.
4. Confirm the cluster to be deleted, then choose **Delete cluster**.

On the cluster list page, the cluster status is updated as the cluster is deleted.

After you complete this tutorial, you can find more information about Amazon Redshift and next steps in Additional resources (p. 35).

Getting started with common database tasks

Following, you can find a description and walkthrough for common tasks so you can begin using Amazon Redshift databases.

After you connect to the initial cluster `dev` database, you can create a new database. Independent of whether you choose to use the sample dataset or bring your own data to Amazon Redshift while creating a cluster, Amazon Redshift creates the `dev` database.

The examples in this section assume the following:

- You have signed up for Amazon Redshift. For more information, see Prerequisites (p. 1).
- You have created an Amazon Redshift cluster. For more information, see Getting started with Amazon Redshift clusters and data loading (p. 6).
- You have established a connection to the cluster from your SQL client tool, such as the Amazon Redshift console query editor. For more information, see Step 3: Grant access to one of the query editors and run queries (p. 13).

Important
The cluster that you deployed for this exercise runs in a live environment. As long as it's running, it accrues charges to your AWS account. For pricing information, see the Amazon Redshift pricing page.
To avoid unnecessary charges, delete your cluster when you are done with it. The final step of the exercise explains how to do so.

Task 1: Create a database

After you verify that your cluster is up and running, you can create your own first database. This database is where you actually create tables, load data, and run queries. A single cluster can host multiple databases. For example, you can have a `SALESDB` database and an `ORDERSDB` database on the same cluster.

For example, to create a database named **`SALESDB`**, run the following command in your SQL client tool.

```
CREATE DATABASE SALESDB;
```

For this exercise, accept the defaults. For information about more command options, see CREATE DATABASE in the *Amazon Redshift Database Developer Guide*.

After you have created the SALESDB database, you can connect to the new database from your SQL client. Use the same connection parameters as you used for your current connection, but change the database name to `SALESDB`.

Task 2: Create a user

By default, only the admin user that you created when you launched the cluster has access to the initial database in the cluster. To grant other users access, create one or more accounts. Database user accounts are global across all the databases in a cluster, and not per individual databases.

Use the CREATE USER command to create a new user. When you create a new user, you specify the name of the new user and a password. We recommend that you specify a password for the user. It must have 8–64 characters, and it must include at least one uppercase letter, one lowercase letter, and one numeral.

For example, to create a user named **`GUEST`** with password **`ABCd4321`**, run the following command.

```
CREATE USER GUEST PASSWORD 'ABCd4321';
```

To connect to the `SALESDB` database as the `GUEST` user, use the same password when you created the user, such as `ABCd4321`.

For information about other command options, see CREATE USER in the *Amazon Redshift Database Developer Guide*.

Task 3: Create a schema

After you create a new database, you can create a new schema in the current database. A *schema* is a namespace that contains named database objects such as tables, views, and user-defined functions (UDFs). A database can contain one or multiple schemas, and each schema belongs to only one database. Two schemas can have different objects that share the same name.

You can create multiple schemas in the same database to organize data the way that you want or to group your data functionally. For example, you can create a schema to store all your staging data and another schema to store all the reporting tables. You can also create different schemas to store data relevant to different business groups that are in the same database. Each schema can store different database objects, such as tables, views, and user-defined functions (UDFs). In addition, you can create schemas with the AUTHORIZATION clause. This clause gives ownership to a specified user or sets a quota on the maximum amount of disk space that the specified schema can use.

Amazon Redshift automatically creates a schema called `public` for every new database. When you don't specify the schema name while creating database objects, the objects go into the `public` schema.

To access an object in a schema, qualify the object by using the `schema_name.table_name` notation. The qualified name of the schema consists of the schema name and table name separated by a dot. For example, you might have a `sales` schema that has a `price` table and an `inventory` schema that also has a `price` table. When you refer to the `price` table, you must qualify it as `sales.price` or `inventory.price`.

The following example creates a schema named **`SALES`** for the user `GUEST`.

```
CREATE SCHEMA SALES AUTHORIZATION GUEST;
```

For information about more command options, see CREATE SCHEMA in the *Amazon Redshift Database Developer Guide*.

To view the list of schemas in your database, run the following command.

```
select * from pg_namespace;
```

The output should look similar to the following.

```
  nspname              | nspowner |         nspacl
-----------------------+----------+--------------------------
  sales                |  100     |
  pg_toast             |   1      |
  pg_internal          |   1      |
  catalog_history      |   1      |
  pg_temp_1            |   1      |
  pg_catalog           |   1      | {rdsdb=UC/rdsdb,=U/rdsdb}
  public               |   1      | {rdsdb=UC/rdsdb,=U/rdsdb}
  information_schema   |   1      | {rdsdb=UC/rdsdb,=U/rdsdb}
```

For more information on how to query catalog tables, see Querying the catalog tables in the *Amazon Redshift Database Developer Guide.*

Use the GRANT statement to give permissions to users for the schemas.

The following example grants privilege to the `GUEST` user to select data from all tables or views in the `SALESCHEMA` using a SELECT statement.

```
GRANT SELECT ON ALL TABLES IN SCHEMA SALES TO GUEST;
```

The following example grants all available privileges at once to the GUEST user.

```
GRANT ALL ON SCHEMA SALES TO GUEST;
```

Task 4: Create a table

After you create your new database, create tables to hold your data. Specify any column information when you create the table.

In the following example, the GUEST user logs on to the SALESDB and creates a new table.

For example, to create a table named **DEMO**, run the following command.

```
CREATE TABLE Demo (
  PersonID int,
  City varchar (255)
);
```

You can also create a table using the schema_name.object_name notation to create the table in the SALES schema.

```
CREATE TABLE SALES.DEMO (
  PersonID int,
  City varchar (255)
);
```

To view and inspect schemas and their tables, you can use the Amazon Redshift query editor. Or you can see the list of tables in schemas using system views. For more information, see Task 6: Query the system tables (p. 22).

By default, new database objects, such as tables, are created in the default schema named public created during cluster creation. You can use another schema to create database objects. For more information about schemas, see Managing database security in the *Amazon Redshift Database Developer Guide*.

The encoding, distkey, and sortkey columns are used by Amazon Redshift for parallel processing. For more information about designing tables that incorporate these elements, see Amazon Redshift best practices for designing tables.

Insert data rows into a table

After you create a table, insert rows of data into that table.

> **Note**
> The INSERT command inserts rows into a table. For standard bulk loads, use the COPY command. For more information, see Use a COPY command to load data.

For example, to insert values into the DEMO table, run the following command.

```
INSERT INTO DEMO VALUES (781, 'San Jose'), (990, 'Palo Alto');
```

Select data from a table

After you create a table and populate it with data, use a SELECT statement to display the data contained in the table. The SELECT * statement returns all the column names and row values for all of the data in

a table. Using SELECT is a good way to verify that recently added data was correctly inserted into the table.

To view the data that you entered in the **DEMO** table, run the following command.

```
SELECT * from DEMO;
```

The result should look like the following.

```
 personid |   city
----------+----------
      781 | San Jose
      990 | Palo Alto
(2 rows)
```

For more information about using the SELECT statement to query tables, see SELECT.

Task 5: Load sample data

Most of the examples in this guide use the TICKIT sample dataset. You can download the file tickitdb.zip, which contains individual sample data files.

You can load the sample data to your own Amazon S3 bucket.

To load the sample data for your database, first create the tables. Then use the COPY command to load the tables with sample data that is stored in an Amazon S3 bucket. For steps to create tables and load sample data, see Step 4: Load data from Amazon S3 to Amazon Redshift (p. 14).

Task 6: Query the system tables

In addition to the tables that you create, your database contains a number of system tables. These system tables contain information about your installation and about the various queries and processes that are running on the system. You can query these system tables to collect information about your database.

> **Note**
> The description for each table in this documentation indicates whether a table is visible to all users or only to superusers. Log in as a superuser to query tables that are visible only to superusers.

Amazon Redshift provides access to the following types of system tables:

- STL tables

  These system tables are generated from Amazon Redshift log files to provide a history of the system. Logging tables have an STL prefix.
- STV tables

  These tables are virtual system tables that contain snapshots of the current system data. Snapshot tables have an STV prefix.
- System views

  System views contain a subset of data found in several of the STL and STV system tables. Systems views have an SVV or SVL prefix.
- System catalog tables

The system catalog tables store schema metadata, such as information about tables and columns. System catalog tables have a PG prefix.

To retrieve system table information about a query, you might need to specify the process ID associated with that query. For more information, see Determine the process ID of a running query (p. 25).

View a list of table names

To view a list of all tables in a schema, you can query the PG_TABLE_DEF system catalog table. You can first examine the setting for `search_path`.

```
SHOW search_path;
```

The result should look similar to the following,

```
  search_path
---------------
 $user, public
(1 row)
```

The following example adds the `SALES` schema to the search path and shows all the tables in the `SALES` schema.

```
set search_path to '$user', 'public', 'sales';

SHOW search_path;
      search_path
------------------------
 "$user", public, sales
(1 row)

select * from pg_table_def where schemaname = 'sales';
 schemaname | tablename |  column  |          type          | encoding | distkey | sortkey
 | notnull
------------+-----------+----------+------------------------+----------+---------+---------
+---------
 sales      | demo      | personid | integer                | az64     | f       |       0
 | f
 sales      | demo      | city     | character varying(255) | lzo      | f       |       0
 | f
(2 rows)
```

The following example shows a list of all tables called `DEMO` in all schemas on the current database.

```
select * from pg_table_def where tablename = 'demo';
 schemaname | tablename |  column  |          type          | encoding | distkey | sortkey
 | notnull
------------+-----------+----------+------------------------+----------+---------+---------
+---------
 public     | demo      | personid | integer                | az64     | f       |       0
 | f
 public     | demo      | city     | character varying(255) | lzo      | f       |       0
 | f
 sales      | demo      | personid | integer                | az64     | f       |       0
 | f
 sales      | demo      | city     | character varying(255) | lzo      | f       |       0
 | f
(4 rows)
```

For more information, see PG_TABLE_DEF.

You can also use the query editor v2 to view all the tables in a specified schema by first choosing a database that you want to connect to.

View users

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

```
SELECT * FROM pg_user;
  usename    | usesysid | usecreatedb | usesuper | usecatupd |  passwd  | valuntil |
 useconfig
------------+----------+-------------+----------+-----------+----------+----------
+-----------
 rdsdb      |        1 | true        | true     | true      | ******** | infinity |
 awsuser    |      100 | true        | true     | false     | ******** |          |
 guest      |      104 | true        | false    | false     | ******** |          |
(3 rows)
```

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

```
SELECT * FROM pg_user WHERE usesysid > 1;
  usename    | usesysid | usecreatedb | usesuper | usecatupd |  passwd  | valuntil |
 useconfig
------------+----------+-------------+----------+-----------+----------+----------
+-----------
 awsuser    |      100 | true        | true     | false     | ******** |          |
 guest      |      104 | true        | false    | false     | ******** |          |
(2 rows)
```

View recent queries

In the previous example, the user ID (USESYSID) for `adminuser` is 100. To list the five most recent queries run by `adminuser`, you can query the SVL_QLOG view.

The SVL_QLOG view is a friendlier subset of information from the STL_QUERY table. You can use this view to find the query ID (QUERY) or process ID (PID) for a recently run query. You can also use this view to check how long it took a query to complete. SVL_QLOG includes the first 60 characters of the query string (SUBSTRING) to help you locate a specific query. Use the LIMIT clause with your SELECT statement to limit the results to five rows.

```
SELECT query, pid, elapsed, substring from svl_qlog
WHERE userid = 100
ORDER BY starttime desc
LIMIT 4;
```

The result look something like the following.

```
 query|  pid  |  elapsed |       substring
------+-------+----------+-----------------------------------------------------------------
 892  | 21046 |  55868   | SELECT query, pid, elapsed, substring from svl_qlog WHERE us
 620  | 17635 | 1296265  | SELECT query, pid, elapsed, substring from svl_qlog WHERE us
 610  | 17607 |  82555   | SELECT * from DEMO;
 596  | 16762 |  226372  | INSERT INTO DEMO VALUES (100);)
```

Determine the process ID of a running query

You might need to find the PID for a query that is still running. For example, you need the PID if you need to cancel a query that is taking too long to run. You can query the STV_RECENTS system table to obtain a list of process IDs for running queries, along with the corresponding query string. If your query returns multiple PIDs, you can look at the query text to determine which PID you need.

To determine the PID of a running query, run the following SELECT statement.

```
SELECT pid, user_name, starttime, query
FROM stv_recents
WHERE status='Running';
```

Task 7: Cancel a query

If you run a query that is taking too long or is consuming excessive cluster resources, cancel the query. For example, create a list of ticket sellers that includes the seller's name and quantity of tickets sold. The following query selects data from the SALES table and USERS table and joins the two tables by matching SELLERID and USERID in the WHERE clause.

```
SELECT sellerid, firstname, lastname, sum(qtysold)
FROM sales, users
WHERE sales.sellerid = users.userid
GROUP BY sellerid, firstname, lastname
ORDER BY 4 desc;
```

The result looks something like the following.

```
 sellerid | firstname | lastname | sum
----------+-----------+----------+------
  48950   |   Nayda   |   Hood   | 184
  19123   |   Scott   | Simmons  | 164
  20029   |    Drew   | Mcguire  | 164
  36791   |  Emerson  | Delacruz | 160
  13567   |   Imani   |   Adams  | 156
  9697    |  Dorian   |    Ray   | 156
  41579   | Harrison  | Durham   | 156
  15591   |  Phyllis  |  Clay    | 152
  3008    |  Lucas    | Stanley  | 148
  44956   |  Rachel   |Villarreal| 148
```

Note
This is a complex query. For this tutorial, you don't need to worry about how this query is constructed.

The previous query runs in seconds and returns 2,102 rows.

Suppose that you forget to put in the WHERE clause.

```
SELECT sellerid, firstname, lastname, sum(qtysold)
FROM sales, users
GROUP BY sellerid, firstname, lastname
ORDER BY 4 desc;
```

The result set includes all of the rows in the SALES table multiplied by all the rows in the USERS table (49989*3766). This is called a Cartesian join, and it isn't recommended. The result is over 188 million rows and takes a long time to run.

To cancel a running query, use the CANCEL command with the query's PID.

To find the process ID, start a new session and query the STV_RECENTS table, as shown in the previous step. The following example shows how you can make the results more readable. To do this, use the TRIM function to trim trailing spaces and show only the first 20 characters of the query string.

```
SELECT pid, trim(user_name), starttime, substring(query,1,20)
FROM stv_recents
WHERE status='Running';
```

The result looks something like the following.

```
  pid   |   btrim    |         starttime          |      substring
-------+------------+----------------------------+----------------------
 610    | adminuser  | 2013-03-28 18:39:49.355918 | select sellerid, fir
(1 row)
```

To cancel the query with PID 18764, run the following command.

```
CANCEL 610;
```

Note
The CANCEL command doesn't stop a transaction. To stop or roll back a transaction, use the ABORT or ROLLBACK command. To cancel a query associated with a transaction, first cancel the query then stop the transaction.

If the query that you canceled is associated with a transaction, use the ABORT or ROLLBACK command to cancel the transaction and discard any changes made to the data:

```
ABORT;
```

Unless you are signed on as a superuser, you can cancel only your own queries. A superuser can cancel all queries.

Cancel a query from another session

If your query tool doesn't support running queries concurrently, start another session to cancel the query. For example, the query editor that we use in the *Amazon Redshift Getting Started Guide* doesn't support multiple concurrent queries. To start another session with the query editor, choose **File**, **New Window** and connect using the same connection parameters. Then you can find the PID and cancel the query.

Cancel a query using the superuser queue

If your current session has too many queries running concurrently, you might not be able to run the CANCEL command until another query finishes. In that case, run the CANCEL command using a different workload management query queue.

By using workload management, you can run queries in different query queues so that you don't need to wait for another query to complete. The workload manager creates a separate queue, called the Superuser queue, that you can use for troubleshooting. To use the Superuser queue, log on a superuser and set the query group to 'superuser' using the SET command. After running your commands, reset the query group using the RESET command.

To cancel a query using the Superuser queue, run these commands.

```
SET query_group TO 'superuser';
CANCEL 610;
RESET query_group;
```

Task 8: Clean up your resources

If you deployed a cluster to complete this exercise, when you are finished with the exercise delete the cluster. Deleting the cluster stops it accruing charges to your AWS account.

To delete the cluster, follow the steps in Deleting a cluster in the *Amazon Redshift Management Guide*.

If you want to keep the cluster, keep the sample data for reference. Most of the examples in this guide use the tables that you create in this exercise. The size of the data won't have any significant effect on your available storage.

If you want to keep the cluster, but want to clean up the sample data, run the following command to drop the `SALESDB` database.

```
DROP DATABASE SALESDB;
```

If you didn't create a `SALESDB` database, or if you don't want to drop the database, run the following commands to drop just the tables.

```
DROP TABLE DEMO;
DROP TABLE users;
DROP TABLE venue;
DROP TABLE category;
DROP TABLE date;
DROP TABLE event;
DROP TABLE listing;
DROP TABLE sales;
```

Getting started with Amazon Redshift Serverless basics

If you are a first-time user of Amazon Redshift Serverless, we recommend that you read the following sections to help you get started using Amazon Redshift Serverless.

Topics

Getting started with the Amazon Redshift Serverless console

In the Amazon Redshift Serverless console, you can manage all of your serverless resources, such as workgroups, namespaces, data backups, and alarms. The first time you log in to the Amazon Redshift Serverless console, you are prompted to access the getting started experience, which you can use to configure your Amazon Redshift Serverless.

Connecting to Amazon Redshift Serverless

To connect to your Amazon Redshift Serverless, use one of the following methods:

- From your preferred SQL client, use the Amazon Redshift provided JDBC driver version 2 driver. For more information, see Connecting to Amazon Redshift Serverless through JDBC drivers.
- To use the Amazon Redshift Data API to connect to Amazon Redshift Serverless, omit the `cluster-identifier` parameter and include `workgroup-name` in AWS CLI calls to route your calls to Amazon Redshift Serverless. For more information about the Data API, see Using the Amazon Redshift Data API.
- To set up a Secure Sockets Layer (SSL) connection to encrypt queries and data, you can use the same configuration you use to set up a connection to a provisioned Redshift cluster. For more information, see Configuring security options for connections.
- You can connect to Amazon Redshift Serverless from an Amazon Redshift managed VPC endpoint as well. For more information, see Connecting to Amazon Redshift Serverless from an Amazon Redshift managed VPC endpoint.
- You can create a publicly accessible Amazon Redshift Serverless to query it from a SQL client in the public internet. For more information, see Creating a publicly accessible Amazon Redshift Serverless and connecting to it.
- You can invoke query editor v2 from the Amazon Redshift Serverless console, a new browser tab opens with the query editor. The query editor v2 connects from your client machine to the Amazon Redshift

Serverless environment. For more information, see Step 2: Query sample data in Amazon Redshift query editor v2 (p. 31).

Getting started with Amazon Redshift Serverless and data loading

To set up and use Amazon Redshift Serverless, set up your serverless data warehouse and create a database. To do so, you need IAM permissions as described in Using identity-based policies (IAM policies) for Amazon Redshift in the *Amazon Redshift Management Guide*. In addition, you must attach a policy similar to the following policy to your IAM role or IAM user.

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

To get started, open the AWS Management Console, choose the Amazon Redshift console, and then choose **Try Amazon Redshift Serverless**.

If you have the correct AWS Identity and Access Management (IAM) permissions, the first time you access the Amazon Redshift Serverless console, you view the **Get started with Amazon Redshift Serverless** page.

Here is where you can **Use default settings** or **Customize settings** to create your namespace, database, and workgroup.

Amazon Redshift Serverless initializes the resources for your AWS account in the current AWS Region. The initialization process can take a few minutes to set up the environment. The Amazon Redshift query editor v2 opens in a new tab for you to start using Amazon Redshift Serverless.

Permissions required to use Amazon Redshift Serverless

When you use Amazon Redshift Serverless, the IAM role you associate to your Amazon Redshift Serverless needs a trust relationship with both `redshift.amazonaws.com` and `redshift-serverless.amazonaws.com` to allow Amazon Redshift to assume permissions on your behalf. The following example shows the policy document in JSON format to set up a trust relationship with Amazon Redshift Serverless.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "redshift-serverless.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*.

The first time you log in to the Amazon Redshift Serverless console, you are prompted to access the getting started experience, which you can use to configure your Amazon Redshift Serverless. You can use the default settings, or you can customize settings such as credentials, security, and audit logging. Part of the getting started experience is creating a workgroup and namespace, which are collections of compute resources and database objects, respectively. For more information about workgroups and namespaces in Amazon Redshift Serverless, see Overview of Amazon Redshift Serverless workgroups and namespaces.

Topics

Using a sample dataset

In this tutorial, you walk through the process to create an Amazon Redshift Serverless by using a sample dataset. Amazon Redshift Serverless automatically loads the sample dataset, such as the tickit dataset, when you are creating a new Amazon Redshift Serverless. You can immediately query the data.

Topics

Step 1: Set up Amazon Redshift Serverless for the first time

The first time you select the **Serverless dashboard**, you walk through the steps to set up Amazon Redshift Serverless. Under **Get started with the serverless experience**, you can choose **Use default settings**. Amazon Redshift Serverless then creates a default namespace and workgroup. This configuration uses the default settings and becomes active when you associate the default workgroup to the default namespace. In this section, you use the default settings, default workgroup, and default namespace.

For more granular control of your setup, choose **Customize settings**.

To configure with default settings:

1. Under **Configuration**, choose **Use default settings**. Amazon Redshift Serverless creates a default namespace with a default workgroup associated to this namespace.

   The following settings under **Namespace** are set to default values.

   - **Database name and password**
   - **Admin user credentials**
   - **Default IAM role**
   - **IAM roles** - If you haven't created any IAM roles, choose **Associate IAM role** to create and associate an IAM role or roles with the namespace. The IAM role you associate with your namespace must include a trust relationship with `redshift-serverless.amazonaws.com` and `redshift.amazonaws.com`.

The **Workgroup** is set to **default**. Additionally, network settings are set to the **default VPC**, the **default** subnet, and the **default** cluster security group.

2. Choose **Save configuration**.
3. After setup completes, choose **Continue** to go to your **Serverless dashboard**.

To configure with customize settings:

1. Under **Configuration**, choose **Customize settings**. Configure the following settings.

   - **Database name** – The name of the initial (default) database to create in the Amazon Redshift Serverless environment. This database is owned by your account and created in the current AWS Region. The name is `dev` and you can't change it.
   - **Admin user credentials** – The user name and password of the administrator of the initial database. This user has ownership permissions for the database.
   - **Virtual private cloud (VPC)** – The name of the VPC where the database is created.
   - **VPC security groups** – These security groups define which subnets and IP ranges can be used in the VPC.
   - **Subnet** – The subnets in the VPC that is associated with the specified database.
   - The AWS owned KMS key is used by default to encrypt your data. Instead of using the AWS owned KMS key, you can **Customize encryption settings** to choose a **KMS key** you manage – The AWS KMS key is used to encrypt resources in Amazon Redshift Serverless.
   - **Audit logging** – The audit log types you want to export.
   - **Permissions** – The IAM role you associate with Amazon Redshift Serverless must include a trust relationship with `redshift-serverless.amazonaws.com` and `redshift.amazonaws.com`.

   The **Workgroup** is set to **default**. Additionally, network settings are set to the **default VPC**, the **default** subnet, and the **default** cluster security group.
2. Choose **Save configuration**.
3. After setup completes, choose **Continue** to go to your **Serverless dashboard**.

Step 2: Query sample data in Amazon Redshift query editor v2

You can manage and query data using query editor v2. The query editor v2 is a full feature web-based SQL client tool to connect to your Amazon Redshift Serverless data. To get set up to use the Amazon Redshift query editor v2, including which permissions are needed, see Configuring your AWS account in the *Amazon Redshift Management Guide*.

Look for the **Query data** button to query data in your Amazon Redshift Serverless with query editor v2. When you invoke query editor v2 from the Amazon Redshift Serverless console, a new browser tab opens with the query editor. The query editor v2 connects from your client machine to the Amazon Redshift Serverless environment.

1. On the list of resources, choose the **Serverless default** workgroup. query editor v2 automatically connects to Amazon Redshift Serverless using temporary credentials.
2. Under the Amazon Redshift Serverless default workgroup, expand the **sample_data_dev** database. There are three sample schemas corresponding to three sample data sets that you can load onto the Amazon Redshift database.
3. In this tutorial, choose the **tickit** schema to initiate the creation of the **sample_data_dev** database, the **tickit** schema, and the sample tables under the **tickit** schema. Amazon Redshift Serverless also loads the **tickit** sales data in the sample tables.

4. You can run the sample queries against the loaded sample data.

For information about query editor v2, see Querying a database using the Amazon Redshift query editor v2 in the *Amazon Redshift Management Guide*.

Bringing your own data to Amazon Redshift Serverless

In this tutorial, you walk through the process to create an Amazon Redshift Serverless and load your own data.

Topics

Step 1: Set up Amazon Redshift Serverless for the first time

To set up Amazon Redshift Serverless for the first time, follow the steps in Step 1: Set up Amazon Redshift Serverless for the first time (p. 30).

Step 2: Query your own data in Amazon Redshift query editor v2

You can query data from different data sources in Amazon Redshift Serverless. Following is a list of data sources you can query data from:

- To query data in Amazon S3, you can load data into an existing Amazon Redshift table from Amazon S3 using query editor v2. For more information, see Loading data from Amazon S3 in the *Amazon Redshift Management Guide*.
- To query data in AWS Glue catalogues that represent databases in your Amazon S3-based data lakes, you can create an external schema to query your data lake without loading any data into Amazon Redshift Serverless. For more information, see Querying a data lake in the *Amazon Redshift Management Guide*.
- To query data in your data stores using federated queries, you can create an external schema from an Amazon RDS PostgresSQL or MySQL database.

  For information on how to get started using federated queries to PostgreSQL, see Getting started with using federated queries to PostgreSQL.

  For more information on how to get started with using federated queries to MySQL, see Getting started with using federated queries to MySQL.

  For information on how to create an external schema in query editor v2, see Creating the external schema.

Getting started with querying data sources outside your Amazon Redshift database

Following, you can find information about how to get started querying data on remote sources, including remote Amazon Redshift clusters. You can also find information about training machine learning (ML) models using Amazon Redshift.

Topics

Getting started querying your data lake

You can use Amazon Redshift Spectrum to query data in Amazon S3 files without having to load the data into Amazon Redshift tables. You can query data in many formats, including Parquet, ORC, RCFile, TextFile, SequenceFile, RegexSerde, OpenCSV, and AVRO. To define the structure of the files in Amazon S3, you create external schemas and tables. Then, you use an external data catalog such as AWS Glue or your own Apache Hive metastore. Changes to either type of data catalog are immediately available to any of your Amazon Redshift clusters.

After your data is registered with an AWS Glue Data Catalog and enabled with AWS Lake Formation, you can query it by using Redshift Spectrum.

Redshift Spectrum resides on dedicated Amazon Redshift servers that are independent of your cluster. Redshift Spectrum pushes many compute-intensive tasks, such as predicate filtering and aggregation, to the Redshift Spectrum layer. Redshift Spectrum also scales intelligently to take advantage of massively parallel processing.

You can partition the external tables on one or more columns to optimize query performance through partition elimination. You can query and join the external tables with Amazon Redshift tables. You can access external tables from multiple Amazon Redshift clusters and query the Amazon S3 data from any cluster in the same AWS Region. When you update Amazon S3 data files, the data is immediately available for queries from any of your Amazon Redshift clusters.

For more information about Redshift Spectrum, including how to work with Redshift Spectrum and data lakes, see Getting started with Amazon Redshift Spectrum in *Amazon Redshift Database Developer Guide*.

Getting started querying data on remote data sources

You can join data from an Amazon RDS database, an Amazon Aurora database, or Amazon S3 with data in your Amazon Redshift database using a federated query. You can use Amazon Redshift to query

operational data directly (without moving it), apply transformations, and insert data into your Redshift tables. Some of the computation for federated queries is distributed to the remote data sources.

To run federated queries, Amazon Redshift first makes a connection to the remote data source. Amazon Redshift then retrieves metadata about the tables in the remote data source, issues queries, and then retrieves the result rows. Amazon Redshift then distributes the result rows to Amazon Redshift compute nodes for further processing.

For information about setting up your environment for federated queries, see one of the following topics in the *Amazon Redshift Database Developer Guide*:

- Getting started with using federated queries to PostgreSQL
- Getting started with using federated queries to MySQL

Getting started accessing data in other Amazon Redshift clusters

Using Amazon Redshift data sharing, you can share live data with high security and greater ease across Amazon Redshift clusters or AWS accounts for read purposes. You can have instant, granular, and high-performance access to data across Amazon Redshift clusters without your needing to manually copy or move it. Your users can see the most up-to-date and consistent information as it's updated in Amazon Redshift clusters.

Amazon Redshift data sharing is especially useful for these use cases:

- Centralizing business-critical workloads – Use a central extract, transform, and load (ETL) cluster that shares data with multiple business intelligence (BI) or analytic clusters. This approach provides read workload isolation and chargeback for individual workloads.
- Sharing data between environments – Share data among development, test, and production environments. You can improve team agility by sharing data at different levels of granularity.

For more information about data sharing, see Getting started data sharing in the *Amazon Redshift Database Developer Guide*.

Getting started training machine learning models with Amazon Redshift data

Using Amazon Redshift machine learning (Amazon Redshift ML), you can train a model by providing the data to Amazon Redshift. Then Amazon Redshift ML creates models that capture patterns in the input data. You can then use these models to generate predictions for new input data without incurring additional costs. By using Amazon Redshift ML, you can train machine learning models using SQL statements and invoke them in SQL queries for prediction. You can continue to improve the accuracy of the predictions by iteratively changing parameters and improving your training data.

Amazon Redshift ML makes it easier for SQL users to create, train, and deploy machine learning models using familiar SQL commands. By using Amazon Redshift ML, you can use your data in Amazon Redshift clusters to train models with Amazon SageMaker. You can then localize the models, and predictions then can be made within an Amazon Redshift database.

For more information about Amazon Redshift ML, see Getting started with Amazon Redshift ML in the *Amazon Redshift Database Developer Guide.*

Additional resources

When you have completed these tutorials, we recommend that you continue to learn more about the concepts introduced in this guide by using the following Amazon Redshift resources:

- Amazon Redshift Management Guide: This guide builds upon this *Amazon Redshift Getting Started Guide*. It provides in-depth information about the concepts and tasks for creating, managing, and monitoring clusters.
- Amazon Redshift Database Developer Guide: This guide also builds upon this *Amazon Redshift Getting Started Guide*. It provides in-depth information for database developers about designing, building, querying, and maintaining the databases that make up your data warehouse.
  - SQL reference: This topic describes SQL commands and function references for Amazon Redshift.
  - System tables and views: This topic describes system tables and views for Amazon Redshift.
- Tutorials for Amazon Redshift: This topic shows tutorials to learn about Amazon Redshift features.
  - Using spatial SQL functions with Amazon Redshift: This tutorial demonstrates how to use some of the spatial SQL functions with Amazon Redshift.
  - Loading data from Amazon S3: This tutorial describes how to load data into your Amazon Redshift database tables from data files in an Amazon S3 bucket.
  - Querying nested data with Amazon Redshift Spectrum: This tutorial describes how to use Redshift Spectrum to query nested data in Parquet, ORC, JSON, and Ion file formats using external tables.
  - Configuring manual workload management (WLM) queues: This tutorial describes how to configure manual workload management (WLM) in Amazon Redshift.
- Feature videos: These videos help you learn about Amazon Redshift features.
  - To learn how to get started with Amazon Redshift, watch the following video: Getting stated with Amazon Redshift.
  - To learn how Amazon Redshift data sharing works, watch the following video: Amazon Redshift data sharing workflow.
  - To learn how Amazon Redshift Machine Learning (ML) works, watch the following video: Amazon Redshift ML.
  - To learn how to monitor, isolate, and optimize your queries using the query monitoring features on the Amazon Redshift console, watch the following video: Query Monitoring with Amazon Redshift.
- What's new: This webpage lists Amazon Redshift new features and product updates.

Document history

The following table describes the important changes for the *Amazon Redshift Getting Started Guide*.

Latest documentation update: June 30, 2021

Change	Description	Release date
Documentation update	Updated the guide to include new sections about getting started with common database tasks, querying your data lake, querying data on remote sources, sharing data, and training machine learning models with Amazon Redshift data.	June 30, 2021
New feature	Updated the guide to describe the new sample load procedure.	June 4, 2021
Documentation update	Updated the guide to remove the original Amazon Redshift console and improve step flow.	August 14, 2020
New console	Updated the guide to describe the new Amazon Redshift console.	November 11, 2019
New feature	Updated the guide to describe the quick-launch cluster procedure.	August 10, 2018
New feature	Updated the guide to launch clusters from the Amazon Redshift dashboard.	July 28, 2015
New feature	Updated the guide to use new node type names.	June 9, 2015
Documentation update	Updated screenshots and procedure for configuring VPC security groups.	April 30, 2015
Documentation update	Updated screenshots and procedures to match the current console.	November 12, 2014
Documentation update	Moved loading data from Amazon S3 information into its own section and moved next steps section into the final step for better discoverability.	May 13, 2014
Documentation update	Removed the Welcome page and incorporated the content into the main Getting Started page.	March 14, 2014
Documentation update	This is a new release of the *Amazon Redshift Getting Started Guide* that addresses customer feedback and service updates.	March 14, 2014
New guide	This is the first release of the *Amazon Redshift Getting Started Guide*.	February 14, 2013