

Relational Database Service

Best Practices

Issue 01
Date 2024-12-30



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2025. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 Overview.....	1
2 RDS for MySQL.....	5
2.1 Migrating Data from Self-Managed MySQL Databases to RDS for MySQL.....	5
2.1.1 Overview.....	5
2.1.2 Resource Planning.....	6
2.1.3 Operation Process.....	7
2.1.4 Cloud Migration.....	8
2.1.4.1 Creating an RDS for MySQL Instance.....	8
2.1.4.2 Creating a Migration Task.....	9
2.1.4.3 Confirming Migration Results.....	11
2.2 Configuring Remote Single-Active DR for an RDS for MySQL Instance Using DRS.....	12
2.2.1 Overview.....	13
2.2.2 Resource Planning.....	13
2.2.3 Operation Process.....	15
2.2.4 Configuring an RDS for MySQL Instance in the Production Center.....	16
2.2.4.1 Creating a VPC and Security Group.....	16
2.2.4.2 Creating an EIP.....	18
2.2.4.3 Creating an RDS for MySQL Instance.....	18
2.2.5 Configuring an RDS for MySQL Instance in the DR Center.....	21
2.2.5.1 Creating a VPC and Security Group.....	21
2.2.5.2 Creating an RDS for MySQL Instance.....	23
2.2.6 Configuring Remote Disaster Recovery.....	25
2.2.6.1 Creating a DRS Disaster Recovery Task.....	25
2.2.6.2 Configuring the Disaster Recovery Task.....	26
2.2.6.3 Performing a Primary/Standby Switchover.....	28
2.3 Migrating MySQL Databases from Other Clouds to RDS for MySQL.....	29
2.3.1 Overview.....	29
2.3.2 Resource Planning.....	30
2.3.3 Operation Process.....	31
2.3.4 Creating a VPC and Security Group.....	31
2.3.5 Creating an RDS for MySQL Instance.....	33
2.3.6 Configuring a MySQL Instance on Another Cloud.....	36
2.3.7 Cloud Migration.....	37

2.3.7.1 Creating a DRS Migration Task.....	37
2.3.7.2 Checking Migration Results.....	39
2.4 Using RDS for MySQL to Set Up WordPress.....	40
2.5 Using RDS for MySQL to Set Up Discuz!.....	48
2.6 Description of innodb_flush_log_at_trx_commit and sync_binlog.....	52
2.7 How Do I Improve the Query Speed of My RDS for MySQL Instance?.....	54
2.8 Handling RDS for MySQL Long Transactions.....	55
2.9 Security Best Practices.....	57
3 RDS for PostgreSQL.....	62
3.1 Creating a Cross-Region DR Relationship for an RDS for PostgreSQL Instance.....	62
3.1.1 Overview.....	62
3.1.2 Resource Planning.....	64
3.1.3 Operation Process.....	65
3.1.4 Preparing an RDS for PostgreSQL Instance in the Production Center.....	66
3.1.5 Preparing an RDS for PostgreSQL Instance in the DR Center.....	69
3.1.6 Configuring Cross-Region Network Connectivity.....	73
3.1.7 Creating a DR Relationship.....	75
3.1.8 Promoting a DR Instance to Primary.....	79
3.1.9 Removing a DR Relationship.....	80
3.1.10 FAQs.....	80
3.2 RDS for PostgreSQL Publications and Subscriptions.....	80
3.3 User-Defined Data Type Conversion.....	84
3.4 Using Client Drivers to Implement Failover and Read/Write Splitting.....	85
3.5 Using PoWA.....	89
3.5.1 Overview.....	89
3.5.2 Performance Metrics.....	90
3.5.2.1 Database Performance Metrics.....	90
3.5.2.2 Instance Performance Metrics.....	93
3.5.3 PoWA Deployment Models.....	96
3.5.3.1 Deploying PoWA for an RDS for PostgreSQL Instance.....	98
3.5.3.2 Configuring Remote Deployment.....	100
3.5.4 Accessing PoWA.....	103
3.6 Best Practices for Using pg_dump.....	104
3.7 Best Practices for Using PgBouncer.....	108
3.8 Security Best Practices.....	111
4 RDS for SQL Server.....	115
4.1 Restoring Data from Backup Files to RDS for SQL Server DB Instances.....	115
4.2 Migrating Data from a Self-Managed SQL Server Database on an ECS to an RDS for SQL Server DB Instance.....	115
4.3 Modifying Parameters of RDS for SQL Server Instances.....	119
4.4 Supporting DMVs.....	121

4.5 Using the Import and Export Function to Migrate Data from a Local Database to an RDS for SQL Server DB Instance.....	122
4.6 Creating a Subaccount of rdsuser.....	126
4.7 Creating tempdb Files.....	130
4.8 Microsoft SQL Server Publication and Subscription.....	139
4.9 Installing a C# CLR Assembly in RDS for SQL Server.....	142
4.10 Creating a Linked Server for an RDS for SQL Server DB Instance.....	146
4.11 Deploying SQL Server Reporting Services (SSRS) on RDS for SQL Server.....	148
4.12 Shrinking an RDS for SQL Server Database.....	150
4.13 Using DAS to Create and Configure Agent Job and DBLink on the Master and Slave Databases for RDS for SQL Server Instances.....	153
4.14 Creating a Job for Scheduled Instance Maintenance	157
4.15 Using Extended Events.....	165

1 Overview

This document describes best practices for working with Relational Database Service (RDS) and provides operational guidelines that you can follow when using this service.

RDS for MySQL Best Practices

Table 1-1 RDS for MySQL best practices

Reference	Description
Migrating Data from Self-Managed MySQL Databases to RDS for MySQL	Describes how to migrate data from self-managed MySQL databases to RDS for MySQL.
Configuring Remote Single-Active DR for an RDS for MySQL Instance Using DRS	Describes how to use DRS to establish a remote single-active DR relationship for an RDS for MySQL instance.
Migrating MySQL Databases from Other Clouds to RDS for MySQL	Describes how to migrate data from MySQL databases on other clouds to RDS for MySQL.
Using RDS for MySQL to Set Up WordPress	Describes how to set up WordPress in a LAMP environment using Huawei Cloud Virtual Private Cloud (VPC), Elastic Cloud Server (ECS), and RDS for MySQL.
Using RDS for MySQL to Set Up Discuz!	Describes how to set up Discuz! in a LAMP environment using Huawei Cloud Virtual Private Cloud (VPC), Elastic Cloud Server (ECS), and RDS for MySQL.

Reference	Description
Description of innodb_flush_log_at_trx_commit and sync_binlog	Describes the impact of the innodb_flush_log_at_trx_commit and sync_binlog parameters on performance and security.
How Do I Improve the Query Speed of My RDS for MySQL Instance?	Describes how to improve the query speed of an RDS for MySQL instance.
Handling RDS for MySQL Long Transactions	Describes how to locate and kill long-running transactions.
Security Best Practices	Provides guidance on RDS for MySQL security configurations.

RDS for PostgreSQL Best Practices

Table 1-2 RDS for PostgreSQL best practices

Reference	Description
Creating a Cross-Region DR Relationship for an RDS for PostgreSQL Instance	Describes how to create a cross-region DR relationship for an RDS for PostgreSQL instance.
RDS for PostgreSQL Publications and Subscriptions	Describes publications and subscriptions of RDS for PostgreSQL.
User-Defined Data Type Conversion	Describes how to customize a data type conversion in RDS for PostgreSQL.
Using Client Drivers to Implement Failover and Read/Write Splitting	Describes how to use client drivers to enable failover and read/write splitting.
Using PoWA	Describes how to use PoWA to monitor the performance of RDS for PostgreSQL instances.
Best Practices for Using pg_dump	Describes how to use pg_dump to back up data.
Best Practices for Using PgBouncer	Describes how to install, configure, and use PgBouncer.
Security Best Practices	Provides guidance on RDS for PostgreSQL security configurations.

RDS for SQL Server Best Practices

Table 1-3 RDS for SQL Server best practices

Reference	Description
Restoring Data from Backup Files to RDS for SQL Server DB Instances	Describes the version restrictions on RDS for SQL Server backup and restoration.
Migrating Data from a Self-Managed SQL Server Database on an ECS to an RDS for SQL Server DB Instance	Describes how to migrate a self-managed SQL Server database on an ECS to an RDS for SQL Server DB instance.
Modifying Parameters of RDS for SQL Server Instances	Describes how to modify parameter templates of RDS for SQL Server DB instances.
Supporting DMVs	Describes how to dynamically manage views through DMV on RDS for SQL Server.
Using the Import and Export Function to Migrate Data from a Local Database to an RDS for SQL Server DB Instance	Describes how to migrate an on-premises SQL Server database to an RDS for SQL Server DB instance.
Creating a Subaccount of rdsuser	Describes the permissions of the rdsuser account and how to create and manage IAM users under the rdsuser account.
Creating tempdb Files	Describes how to create tempdb temporary data files on RDS for SQL Server.
Microsoft SQL Server Publication and Subscription	Describes how RDS for SQL Server provides the subscription function.
Installing a C# CLR Assembly in RDS for SQL Server	Describes how to add a c#CLR assembly on RDS for SQL Server.
Creating a Linked Server for an RDS for SQL Server DB Instance	Describes how to create a linked server to access another RDS for SQL Server DB instance.
Deploying SQL Server Reporting Services (SSRS) on RDS for SQL Server	Describes how to deploy SQL Server Reporting Services (SSRS) in RDS for SQL Server.
Shrinking an RDS for SQL Server Database	Describes how to use a stored procedure to shrink the size of the data and log files in a specified RDS for SQL Server database.

Reference	Description
Using DAS to Create and Configure Agent Job and DBLink on the Master and Slave Databases for RDS for SQL Server Instances	Describes how to use DAS to create and configure agent jobs and DBLink on primary and standby RDS for SQL Server DB instances.
Creating a Job for Scheduled Instance Maintenance	Describes how to create a scheduled SQL agent job to re-create indexes, update statistics, and shrink the database.

2 RDS for MySQL

2.1 Migrating Data from Self-Managed MySQL Databases to RDS for MySQL

2.1.1 Overview

Scenarios

This chapter includes the following content:

- How to migrate data from self-managed MySQL databases to RDS for MySQL instances

RDS for MySQL Advantages

- **More Services at Lower Costs**
You pay for only RDS instances. There is no hardware or management investment needed.
- **Ultimate User Experience**
 - Fully compatible with MySQL
 - Excellent performance for high concurrency
 - Support for a great number of connections and quicker response
- **High Security**
 - End-to-end database security, including network isolation, access control, transmission encryption, storage encryption, and anti-DDoS
 - Highest-level certification by the NIST-CSF, with 108 key security capabilities
- **High Reliability**
Multiple deployment and DR solutions, including data backup, data restoration, dual-host hot standby, remote DR, and intra-city DR

Service List

- Virtual Private Cloud (VPC)
- Elastic Cloud Server (ECS)
- RDS
- Data Replication Service (DRS)

Notes on Usage

- The resource planning in this best practice is for demonstration only. Adjust it as needed.
- All settings in this best practice are for reference only. For more information about MySQL migration, see [From MySQL to MySQL](#).

Prerequisites

- You have registered with Huawei Cloud.
- Your account balance is greater than or equal to \$0 USD.

2.1.2 Resource Planning

Table 2-1 Resource planning description

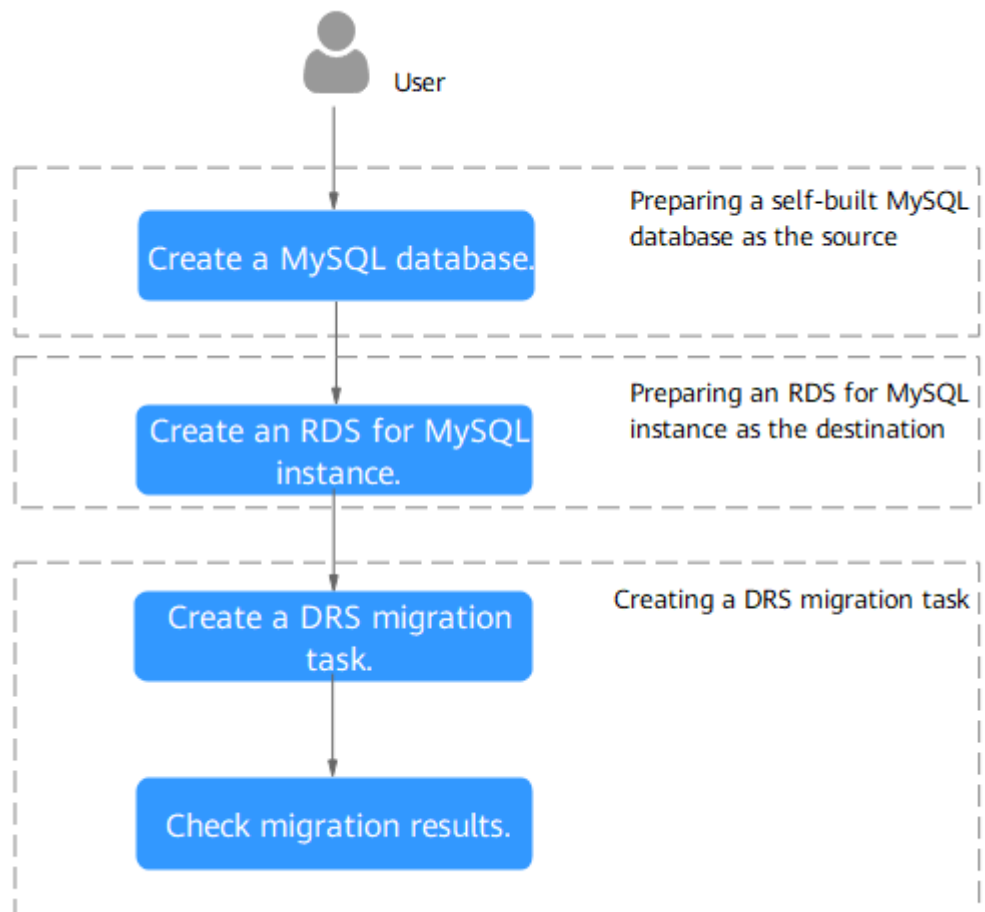
Category	Subcategory	Planned Value	Remarks
RDS	RDS instance name	rds-mysql	Customize a name for easy identification.
	DB engine version	MySQL 5.7	-
	Instance type	Single	In this practice, select a single instance. To improve service reliability, selecting a primary/standby instance is recommended.
	Storage type	Cloud SSD	-
	AZ	AZ3	In this practice, select a single instance. To improve service reliability, create a primary/standby instance and then deploy them in two different AZs.
	Specifications	General-purpose 4 vCPUs 8 GB	-

Category	Subcategory	Planned Value	Remarks
DRS migration task	Task name	DRS-mysql	Custom
	Source DB engine	MySQL	In this practice, the source is a MySQL database built on an ECS.
	Destination DB engine	MySQL	In this practice, the destination is an RDS for MySQL instance.
	Network type	VPC	In this practice, select the VPC network.

2.1.3 Operation Process

The following figure shows the process of creating a MySQL database on an ECS, buying an RDS for MySQL instance, and migrating data from the MySQL database to the RDS instance.

Figure 2-1 Flowchart



2.1.4 Cloud Migration

2.1.4.1 Creating an RDS for MySQL Instance

Create an RDS for MySQL instance that is in the same VPC and security group as the self-managed MySQL database.

Step 1 Go to the [Buy DB Instance](#) page.

Step 2 Configure basic information for the instance. Select **CN-Hong Kong** for **Region**.

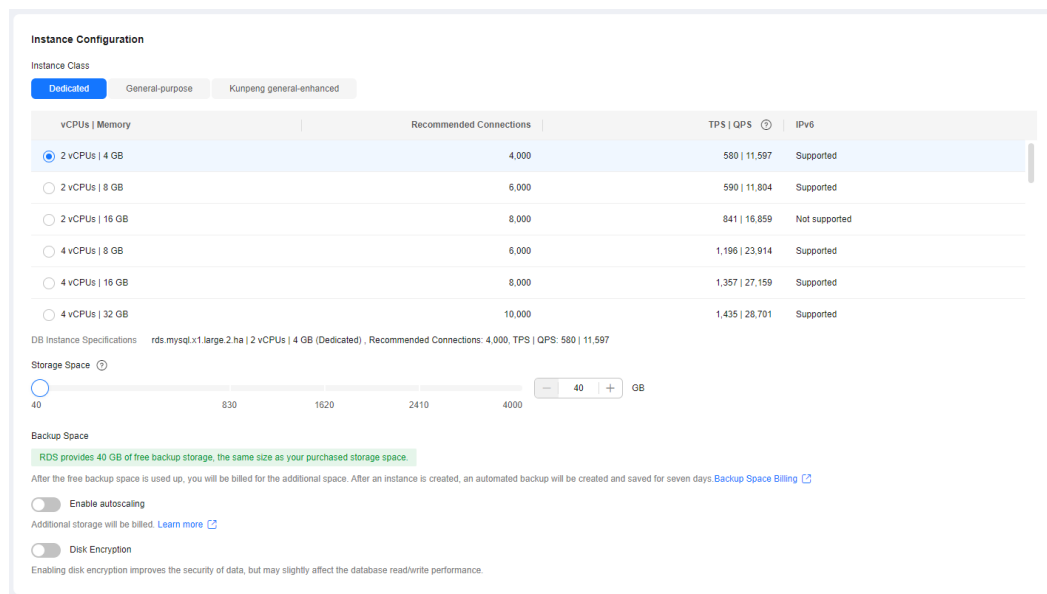
Figure 2-2 Basic information

The screenshot shows the 'Basic Settings' configuration page for an RDS instance. It is divided into two tabs: 'Quick Config' and 'Custom Config', with 'Custom Config' selected. The page is organized into several sections:

- Basic Settings:** Includes 'Billing Mode' with 'Pay-per-use' selected, and a 'Region' dropdown menu.
- Engine Options:** Features 'DB Engine' buttons for GaussDB(for MySQL), MySQL (selected), PostgreSQL, Microsoft SQL Server, and MariaDB. Below this is a 'DB Engine Version' dropdown set to 8.0.
- DB Instance Type:** Offers 'Primary/Standby' (selected) and 'Single' options. A note states: 'Primary/Standby HA architecture is suitable for production databases in large- and medium-sized enterprises, or for applications in Internet, IoT, retail e-commerce, logistics, and gaming industries.'
- Storage Type:** Provides 'Cloud SSD' (selected) and 'Extreme SSD' options.
- Primary AZ:** Lists 'cn-north-4a' (selected), AZ1, cn-north-4b, and cn-north-4c.
- Standby AZ:** Lists cn-north-4a, AZ1 (selected), cn-north-4b, and cn-north-4c.

Step 3 Select an instance class and retain the default values for other parameters.

Figure 2-3 Instance class



Step 4 Click **Next**.

Step 5 Confirm the settings.

- To modify your settings, click **Previous**.
- If you do not need to modify your settings, click **Submit**.

Step 6 Return to the instance list.

If the instance status becomes available, the instance has been created.

----End

2.1.4.2 Creating a Migration Task

This topic describes how to create a DRS migration task to migrate the **loadtest** database from the self-managed MySQL server to an RDS for MySQL instance.

Pre-migration Check

Before creating a migration task, check the migration environment to ensure smooth migration.

This example describes how to migrate data from a self-managed MySQL database to an RDS for MySQL instance. For more information, see [From MySQL to MySQL](#).

Procedure

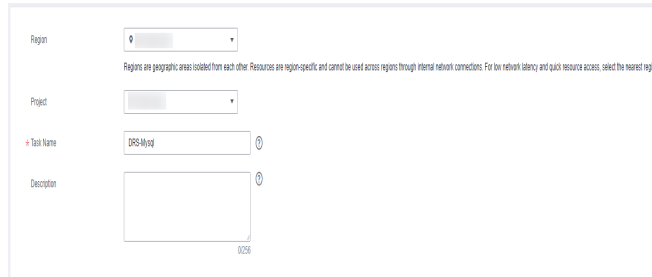
Migrate the **loadtest** database from a self-managed MySQL server to an RDS for MySQL instance.

Step 1 Go to the [Create Migration Task](#) page.

Step 2 Configure parameters as needed.

1. Specify a migration task name. Select **CN-Hong Kong** for **Region**.

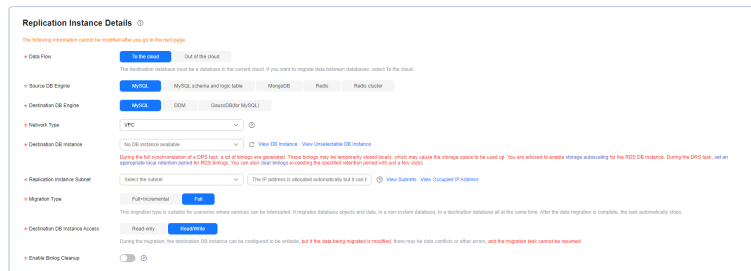
Figure 2-4 Migration task



2. Configure replication instance information.

Select the instance created in **Creating an RDS for MySQL Instance** as the destination instance.

Figure 2-5 Replication instance details



3. Select **default** for **Enterprise Project**.

Step 3 Click **Create Now**.

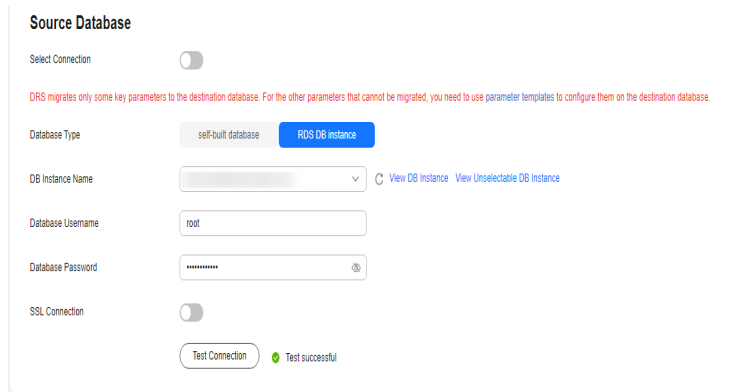
It takes about 5 to 10 minutes to create a replication task.

Step 4 Configure task information and click **Next**.

1. Configure source database information.
2. Click **Test Connection**.

If a successful test message is returned, login to the source is successful.

Figure 2-6 Source database settings



3. Specify a username and password for the destination database.
4. Click **Test Connection**.
If a successful test message is returned, login to the destination is successful.

Figure 2-7 Destination database settings

The screenshot shows a 'Destination Database' configuration form. It contains the following fields and options:

- DB Instance Name:** A text input field with a greyed-out value.
- Database Username:** A text input field containing 'root'.
- Database Password:** A password input field with masked characters and a show/hide icon.
- Migrate Definer to User:** Radio buttons for 'Yes' (selected) and 'No'.
- SSL Connection:** A toggle switch that is currently turned off.
- Test Connection:** A button that has been clicked, resulting in a green checkmark and the text 'Test successful'.

Step 5 On the **Set Task** page, select the accounts and objects to be migrated, and click **Next**.

Select **All** for **Migration Object**.

Step 6 On the **Check Task** page, check the migration task.

If the check is complete and the check success rate is 100%, click **Next**.

Step 7 On the **Compare Parameters** page, click **Next** in the lower right corner to skip the comparison.

Step 8 On the **Confirm Task** page, specify **Start Time**, **Send Notifications**, **SMN Topic**, **Delay Threshold (s)**, and **Stop Abnormal Tasks After**, confirm that the configured information is correct, and click **Submit** to submit the task.

Step 9 After the task is submitted, view and manage it on the **Online Migration Management** page.

----End

2.1.4.3 Confirming Migration Results

You can check migration results with either of the following methods:

Automatic: [Viewing Migration Results on the DRS Console](#). DRS automatically compares migration objects, users, and data of source and destination databases and provides migration results.

Manual: [Viewing Migration Results on the RDS Console](#). You can log in to the destination instance to check whether the databases, tables, and data are migrated.

Viewing Migration Results on the DRS Console


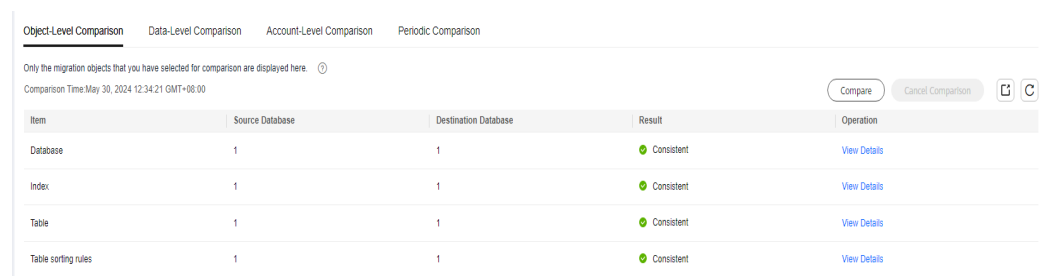
- Step 1** Log in to the [management console](#).
- Step 2** Click  in the upper left corner and select **CN-Hong Kong**.
- Step 3** Click the service list icon on the left and choose **Databases > Data Replication Service**.
- Step 4** Locate the required DRS instance and click its name.
- Step 5** Click **Migration Comparison**.

Figure 2-8 Migration comparison




Item	Source Database	Destination Database	Result	Operation
Database	1	1	Consistent	View Details
Index	1	1	Consistent	View Details
Table	1	1	Consistent	View Details
Table sorting rules	1	1	Consistent	View Details

- Step 6** Select **Compare Data - Validate All Rows/Values** and **Compare Data -Double Check During Cutover** to check whether the objects of the source database have been migrated to the destination database.

If any check fails, rectify the fault by referring to [Solutions to Failed Check Items](#).

----End

Viewing Migration Results on the RDS Console

- Step 1** Log in to the [management console](#).
- Step 2** Click  in the upper left corner and select **CN-Hong Kong**.
- Step 3** Click the service list icon on the left and choose **Databases > Relational Database Service**.
- Step 4** Locate the required RDS instance and click **Log In** in the **Operation** column.
- Step 5** In the displayed dialog box, enter the password and click **Test Connection**.
- Step 6** After the connection test is successful, click **Log In**.
- Step 7** Check and confirm the destination database name and table name. Check whether the data migration is complete.

----End

2.2 Configuring Remote Single-Active DR for an RDS for MySQL Instance Using DRS

2.2.1 Overview

Scenarios

This best practice involves two tasks:

- Create an RDS for MySQL instance.
- Use DRS to establish a remote single-active DR relationship for the RDS for MySQL instance.

Prerequisites

- You have registered with Huawei Cloud.
- Your account balance is at least \$0 USD.

How Cross-Region DR Works

RDS for MySQL instances are deployed in the production and DR data centers. DRS replicates data from the production center to the DR center, keeping data synchronous between your primary instance and the DR instance.

Service List

- Virtual Private Cloud (VPC)
- Elastic IP (EIP)
- Relational Database Service (RDS)
- Data Replication Service (DRS)

Notes on Usage

- The resource planning in this best practice is for demonstration only. Adjust it as needed.
- All settings in this best practice are for reference only. For more information about RDS for MySQL instance DR, see [From MySQL to MySQL \(Single-Active DR\)](#).

2.2.2 Resource Planning

Table 2-2 Resource planning

Category	Subcategory	Planned Value	Description
VPC in the production center	VPC name	vpc-01	Specify a name that is easy to identify.
	Region	CN-Hong Kong	To achieve lower network latency, select the region nearest to you.
	AZ	AZ2	-

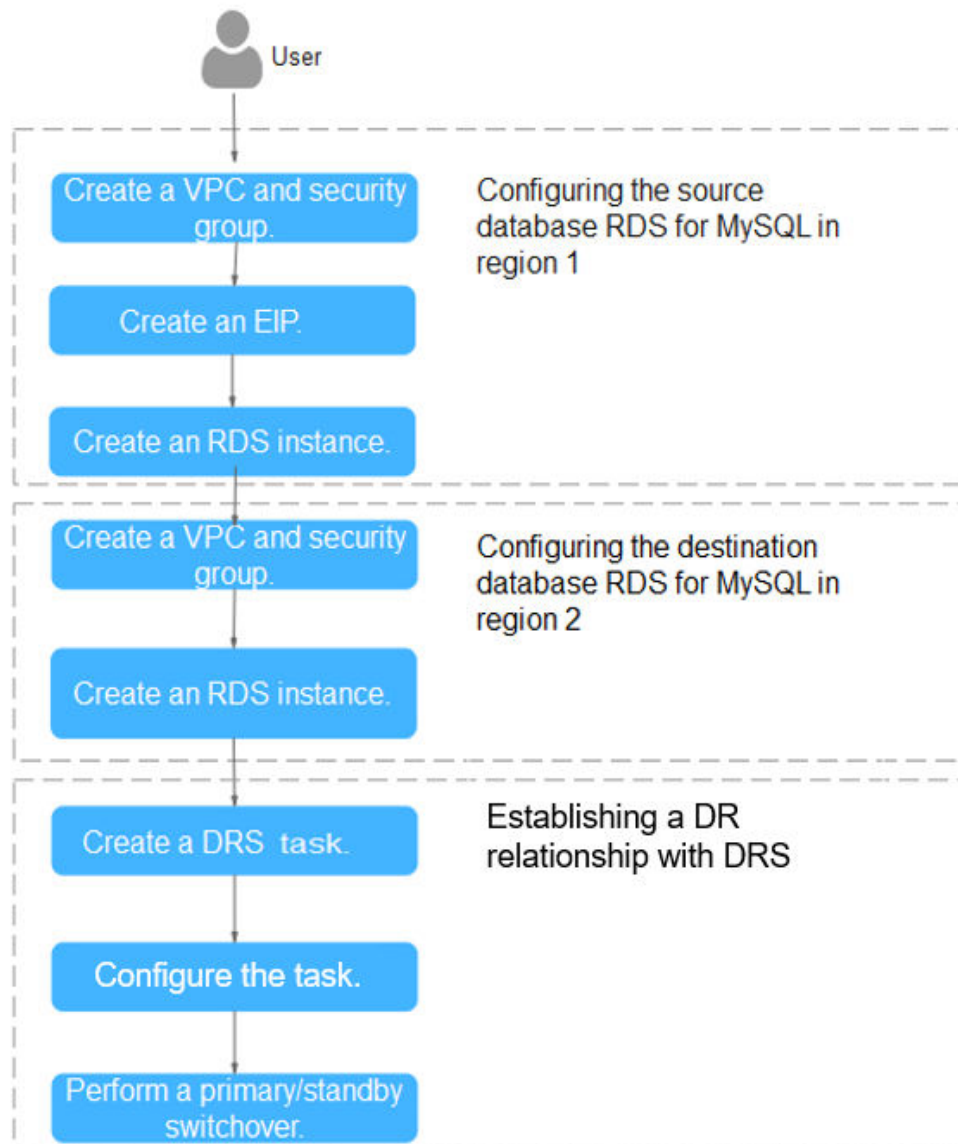
Category	Subcategory	Planned Value	Description
	Subnet	192.168.0.0/24	Select a subnet with sufficient network resources.
	Subnet name	subnet-3c29	Specify a name that is easy to identify.
VPC in the DR center	VPC name	vpc-DR	Specify a name that is easy to identify.
	Region	AP-Singapore	To achieve lower network latency, select the region nearest to you.
	AZ	AZ1	-
	Subnet	192.168.0.0/24	Select a subnet with sufficient network resources.
	Subnet name	subnet-ac27	Specify a name that is easy to identify.
RDS for MySQL instance in the production center	Instance name	rds-database-01	Specify a name that is easy to identify.
	Region	CN-Hong Kong	To achieve lower network latency, select the region nearest to you.
	DB engine version	MySQL 8.0	-
	Instance type	Single	A single instance is used in this example. To improve service reliability, select a primary/standby instance.
	Storage type	Ultra-high I/O	-
	AZ	AZ2	AZ2 is selected in this example. To improve service reliability, select the primary/standby instance type and deploy the primary and standby instances in different AZs.
	Instance specifications	General-enhanced 2 vCPUs 4 GB	-

Category	Subcategory	Planned Value	Description
RDS for MySQL instance in the DR center	Instance name	rds-DR	Specify a name that is easy to identify.
	Region	AP-Singapore	To achieve lower network latency, select the region nearest to you.
	DB engine version	MySQL 8.0	-
	Instance type	Single	A single instance is used in this example. To improve service reliability, select a primary/standby instance.
	Storage type	Cloud SSD	-
	AZ	AZ1	AZ1 is selected in this example. To improve service reliability, select the primary/standby instance type and deploy the primary and standby instances in different AZs.
	Instance specifications	General-purpose 2 vCPUs 8 GB	-
DRS DR task	DR task name	DRS-DR-Task	Specify a name that is easy to identify.
	Source DB engine	MySQL	In this example, the primary instance created in CN-Hong Kong is used as the source database.
	Destination DB engine	MySQL	In this example, the DR instance created in AP-Singapore is used as the destination database.
	Network type	Public network	Public network is used in this example.

2.2.3 Operation Process

You can create a single RDS instance and a DR instance and migrate data from the single instance to the DR instance.

Figure 2-9 Flowchart



2.2.4 Configuring an RDS for MySQL Instance in the Production Center

2.2.4.1 Creating a VPC and Security Group

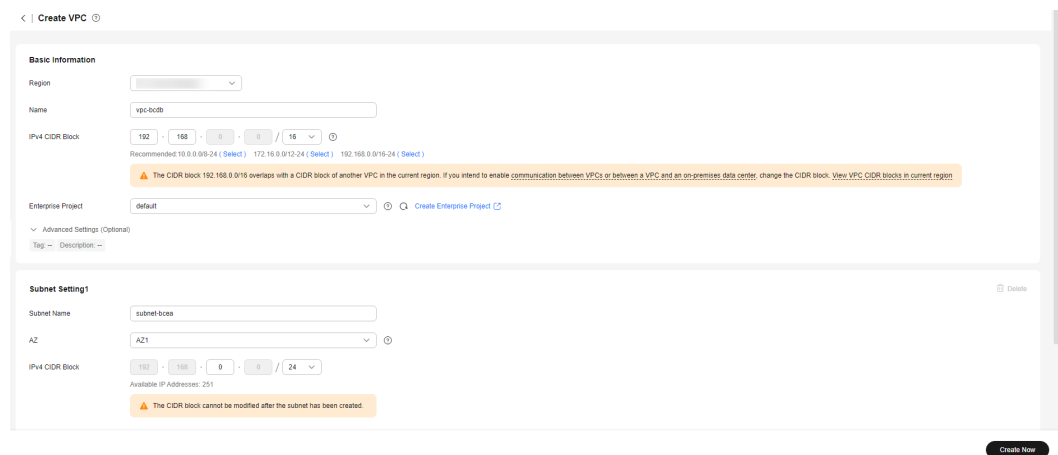
Create a VPC and security group for a DB instance in the production center.

Creating a VPC

Step 1 Go to the [Create VPC](#) page.

Step 2 Configure the basic information, subnet, and IP address. Select **CN-Hong Kong** for **Region**.

Figure 2-10 Creating a VPC




Step 3 Click **Create Now**.

----End

Creating a Security Group

Step 1 Log in to the [management console](#).

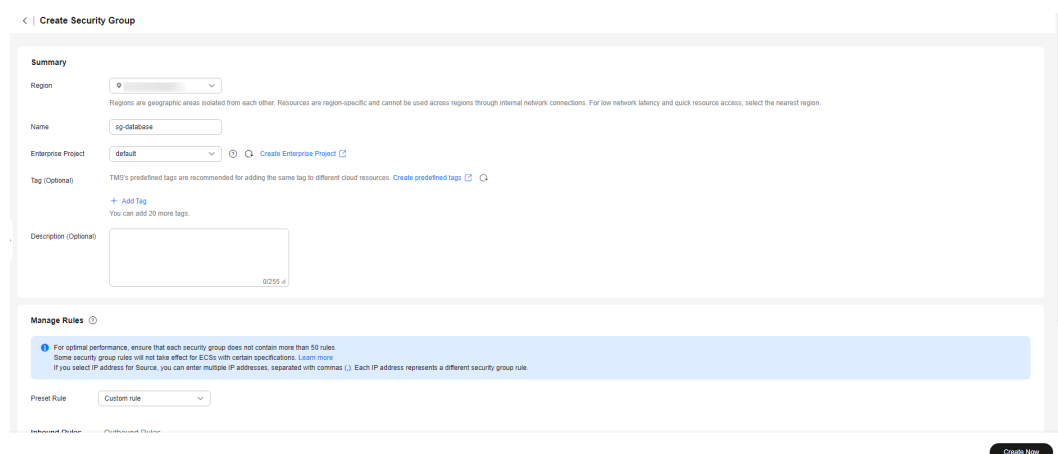
Step 2 Click  in the upper left corner and select **CN-Hong Kong**.

Step 3 Under the service list, choose **Networking > Virtual Private Cloud**.

Step 4 In the navigation pane on the left, choose **Access Control > Security Groups**.

Step 5 Click **Create Security Group**.

Figure 2-11 Creating a security group



Step 6 Click **Create Now**.

----End

2.2.4.2 Creating an EIP

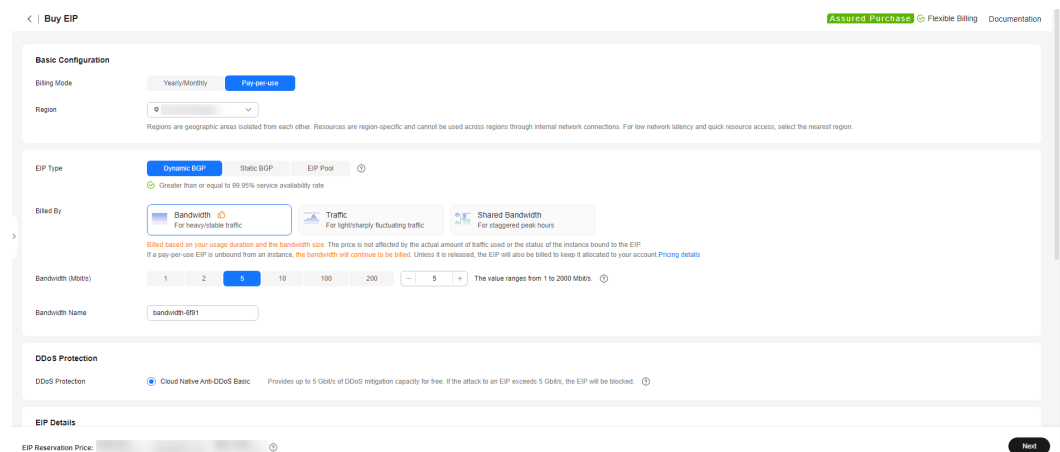
Create an EIP for your source DB instance. Using the EIP, external systems can access your application and DRS can connect to the source DB instance.

Procedure

Step 1 Go to the [Buy EIP](#) page.

Step 2 Configure required parameters. Select **CN-Hong Kong** for **Region**.

Figure 2-12 Buying an EIP



Step 3 Click **Next**.

Step 4 Confirm the information and click **Submit**.

----End

2.2.4.3 Creating an RDS for MySQL Instance

Create an RDS for MySQL instance (source database), and select the VPC and EIP you configured for the instance.

Procedure

Step 1 Go to the [Buy DB Instance](#) page.

Step 2 Select **CN-Hong Kong** for **Region**. Configure instance information and click **Buy**.

Figure 2-13 Selecting a DB engine

Quick Config Custom Config

Basic Settings

Billing Mode ⓘ
Yearly/Monthly **Pay-per-use**

Region
▼
Regions are geographic areas isolated from each other. For low network latency and quick resource access, select the nearest region.

Engine Options

DB Engine
GaussDB(for MySQL) **MySQL** PostgreSQL Microsoft SQL Server MariaDB

DB Engine Version
8.0 ▼

DB Instance Type
Primary/Standby Single
Primary/standby HA architecture is suitable for production databases in large- and medium-sized enterprises, or for applications in Internet, IoT, retail e-commerce, logistics, and gaming industries.

Storage Type
Cloud SSD Extreme SSD

Primary AZ
cn-north-4a AZ1 cn-north-4b cn-north-4c
Multi-AZ deployment provides disaster recovery capabilities across AZs.

Standby AZ
cn-north-4a **AZ1** cn-north-4b cn-north-4c

Figure 2-14 Selecting specifications

Instance Configuration

Instance Class
Dedicated General-purpose Kumpeng general-enhanced

vCPUs Memory	Recommended Connections	TPS QPS ⓘ	IPv6
<input checked="" type="radio"/> 2 vCPUs 4 GB	4,000	580 11,597	Supported
<input type="radio"/> 2 vCPUs 8 GB	6,000	590 11,804	Supported
<input type="radio"/> 2 vCPUs 16 GB	8,000	841 16,859	Not supported
<input type="radio"/> 4 vCPUs 8 GB	6,000	1,196 23,914	Supported
<input type="radio"/> 4 vCPUs 16 GB	8,000	1,357 27,159	Supported
<input type="radio"/> 4 vCPUs 32 GB	10,000	1,435 28,701	Supported

DB Instance Specifications: rds.mysql.x1.large.2.ha | 2 vCPUs | 4 GB (Dedicated), Recommended Connections: 4,000, TPS | QPS: 580 | 11,597

Storage Space ⓘ
40 830 1620 2410 4000 - 40 + GB


Backup Space
RDS provides 40 GB of free backup storage, the same size as your purchased storage space.
After the free backup space is used up, you will be billed for the additional space. After an instance is created, an automated backup will be created and saved for seven days [Backup Space Billing](#) ⓘ

Enable autoscaling
Additional storage will be billed. [Learn more](#) ⓘ

Disk Encryption
Enabling disk encryption improves the security of data, but may slightly affect the database read/write performance.

Figure 2-15 Configuring network information as planned

Basic Settings


DB Instance Name 


If you buy multiple DB instances at a time, they will be named with four digits appended in the format "DB instance name-SN". For example, if the DB instance name is "instance", the first instance will be named "instance-0001", the second "instance-0002", and so on.

Password

To log in, you will have to reset the password later on the Basic Information page for this instance.


Connectivity

VPC 

  [Create VPC](#)

The VPC an RDS instance is deployed in cannot be changed later. ECSs in different VPCs cannot communicate with each other by default.

Subnet


 


IPv6 CIDR block: 2407:c080:1200:217e::/64
An EIP is required if you want to access DB instances through a public network. [View EIP](#)

IPv4 Address

Addresses available: 251 [View in-use IP Addresses](#)

Database Port

Security Group 

  [View Security Group](#)

Ensure that port 3306 of the security group allows traffic from your server IP address to the DB instance. [Create Security Group](#)




Security Group Rules 


Figure 2-16 Additional options


Additional Options

Enterprise Project 


  [Create Enterprise Project](#)

Parameter Template


  [View Parameter Template](#)

 Using a high-performance template tends to result in lost data and replication exceptions after an instance recovers from a crash. There may also be out of memory (OOM) errors with small instance classes in high concurrency scenarios. For details, see the user guide. [View Details](#)

Time Zone

Table Name 

Tag

TMS's predefined tags are recommended for adding the same tag to different cloud resources. [Create predefined tags](#) 



[+ Add Tag](#)

You can add 20 more tags.

Read Replica

Required Duration and Quantity

Quantity

You can create 50 more instances (read replicas included). [Increase Quota](#)

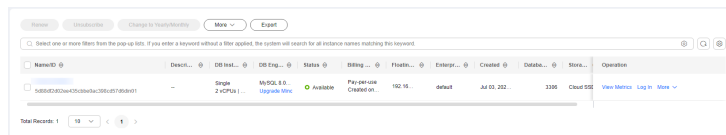
Step 3 Confirm the settings.

- To modify your settings, click **Previous**.
- If there is no need to modify your settings, click **Submit**.

Step 4 Bind an EIP to the created instance.

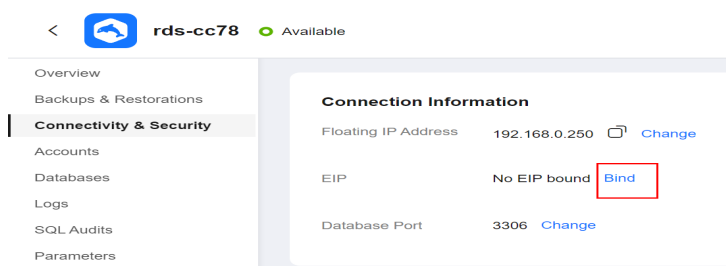
1. On the **Instances** page, click the instance name to go to the **Overview** page.

Figure 2-17 Locating your instance in the list



2. In the navigation pane on the left, choose **Connectivity & Security**. In the **Connection Information** area, click **Bind** next to the **EIP** field.
3. In the displayed dialog box, all unbound EIPs are listed. Select the EIP you have created for the instance and click **Yes**.

Figure 2-18 Binding an EIP



----End

2.2.5 Configuring an RDS for MySQL Instance in the DR Center

2.2.5.1 Creating a VPC and Security Group

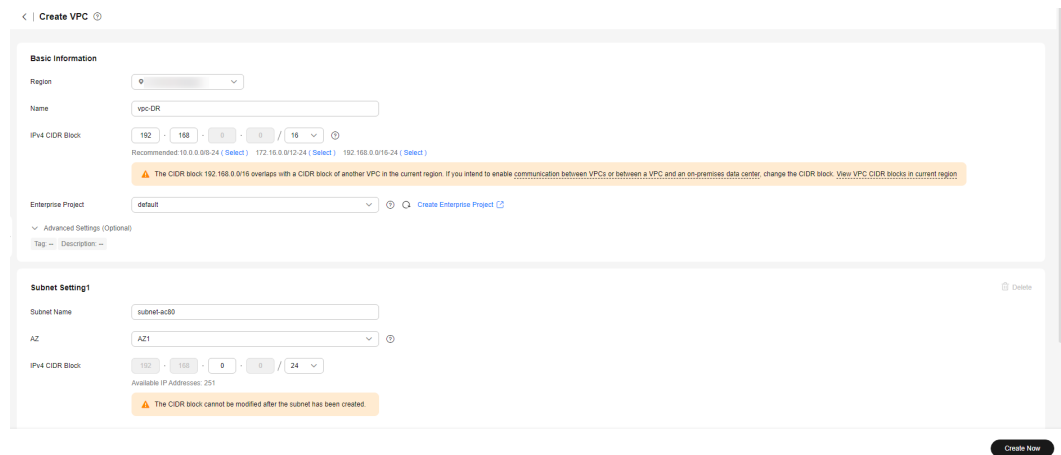
Create a VPC and security group for the DR instance to be configured, ensuring that it is in a different region from the instance created for production center.

Creating a VPC

Step 1 Go to the **Create VPC** page.

Step 2 Configure the basic information, subnet, and IP address. Select **AP-Singapore** for **Region**.

Figure 2-19 Creating a VPC




Step 3 Click **Create Now**.

----End

Creating a Security Group

Step 1 Log in to the [management console](#).

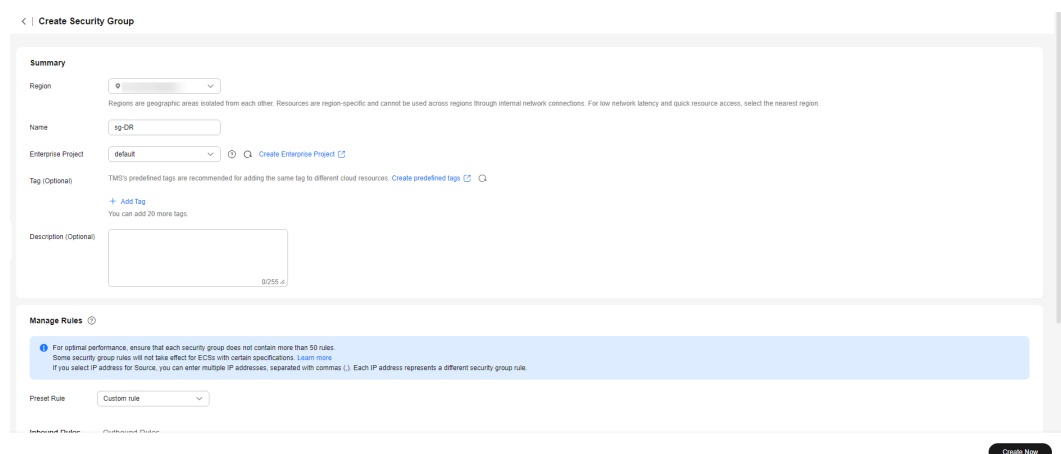
Step 2 Click  in the upper left corner of the management console and select **AP-Singapore**.

Step 3 Under the service list, choose **Networking > Virtual Private Cloud**.

Step 4 In the navigation pane on the left, choose **Access Control > Security Groups**.

Step 5 Click **Create Security Group**.

Figure 2-20 Creating a security group



Step 6 Click **Create Now**.

----End

2.2.5.2 Creating an RDS for MySQL Instance

Create an RDS for MySQL instance as a DR instance and select the VPC you configured for the instance.

Procedure

- Step 1** Go to the [Buy DB Instance](#) page.
- Step 2** Select **AP-Singapore** for **Region**. Configure instance information and click **Buy**.

Figure 2-21 Selecting a DB engine

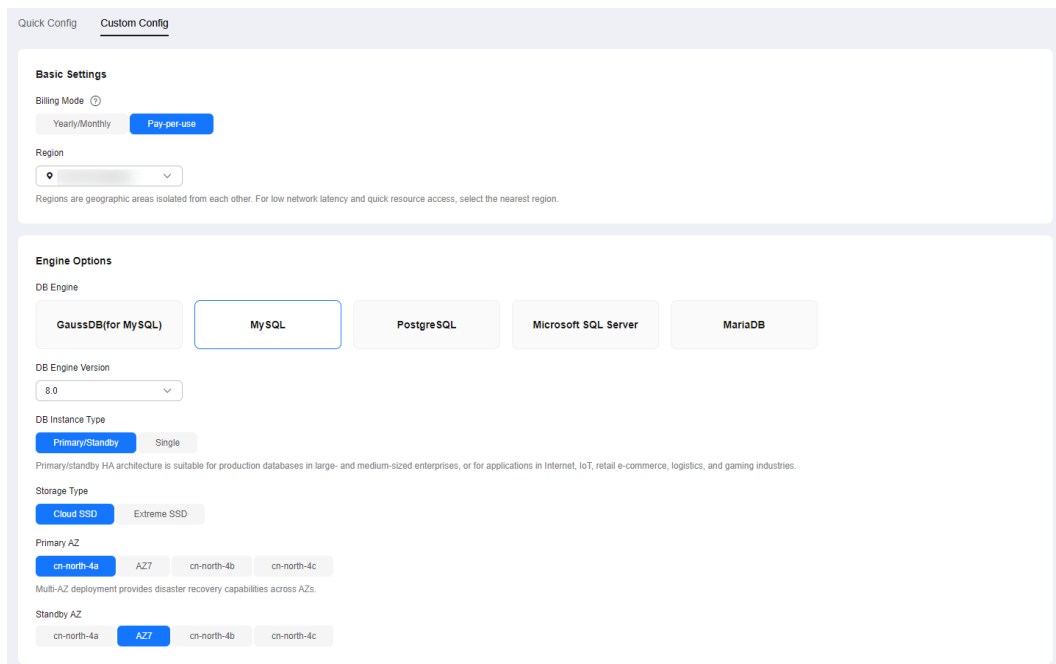


Figure 2-22 Selecting specifications

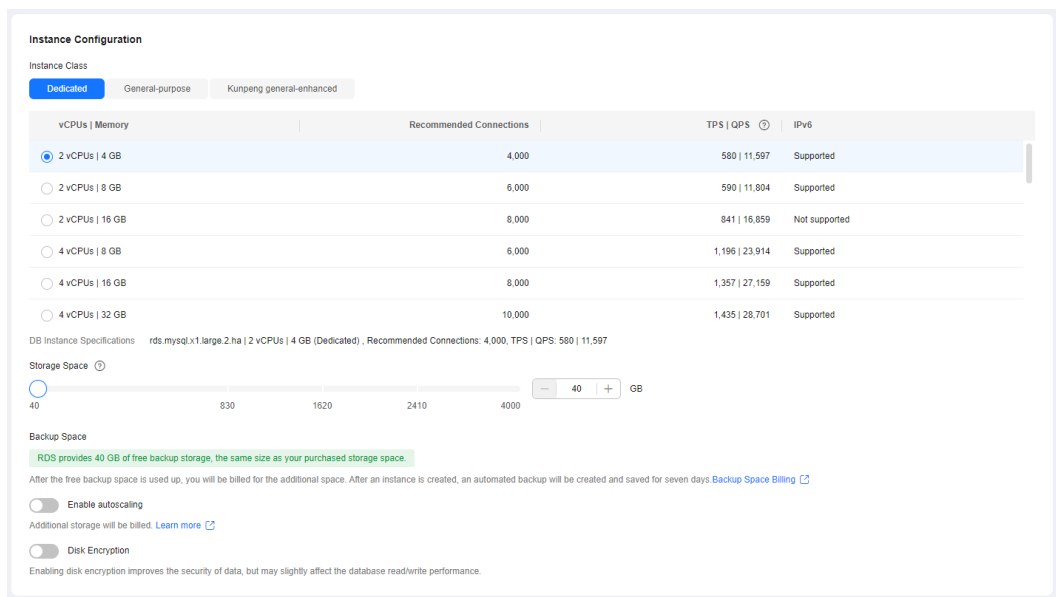


Figure 2-23 Configuring network information as planned

Basic Settings

DB Instance Name ⓘ
rds-9a78

If you buy multiple DB instances at a time, they will be named with four digits appended in the format "DB instance name-SN". For example, if the DB instance name is "instance", the first instance will be named "instance-0001", the second "instance-0002", and so on.

Password

To log in, you will have to reset the password later on the Basic Information page for this instance.

Connectivity

VPC ⓘ
vpc-dr

The VPC an RDS instance is deployed in cannot be changed later. ECSs in different VPCs cannot communicate with each other by default.

Subnet
subnet-mysql(192.168.0.0/24)

An EIP is required if you want to access DB instances through a public network. [View EIP](#)

IPv4 Address
. . .

Addresses available: 251 [View In-use IP Addresses](#)

Database Port
Default port: 3306

Security Group ⓘ
sg-dr

Ensure that port 3306 of the security group allows traffic from your server IP address to the DB instance. [Create Security Group](#)

Security Group Rules ^

Figure 2-24 Additional options

The screenshot shows the 'Additional Options' configuration page. It includes the following sections:

- Enterprise Project:** A dropdown menu set to 'default' with a 'Create Enterprise Project' link.
- Parameter Template:** A dropdown menu set to 'Default-HighPerformance-Edition-MySQL-8.0' with a 'View Parameter Template' link.
- Warning:** A yellow warning box stating: 'Using a high-performance template tends to result in lost data and replication exceptions after an instance recovers from a crash. There may also be out of memory (OOM) errors with small instance classes in high concurrency scenarios. For details, see the user guide. View Details'.
- Time Zone:** A dropdown menu set to '(UTC+08:00) Beijing, Chongqing, Hong Kong, Urumqi'.
- Table Name:** Radio buttons for 'Case sensitive' and 'Case insensitive', with 'Case insensitive' selected.
- Tag:** A section with a '+ Add Tag' link and the text 'You can add 20 more tags.' and a 'Create predefined tags' link.
- Read Replica:** 'Skip' and 'Create' buttons.
- Required Duration and Quantity:** A 'Quantity' section with a numeric input set to '1' and a 'You can create 50 more instances (read replicas included). Increase Quota' link.

Step 3 Confirm the settings.

- To modify your settings, click **Previous**.
- If there is no need to modify your settings, click **Submit**.

----End

2.2.6 Configuring Remote Disaster Recovery

2.2.6.1 Creating a DRS Disaster Recovery Task

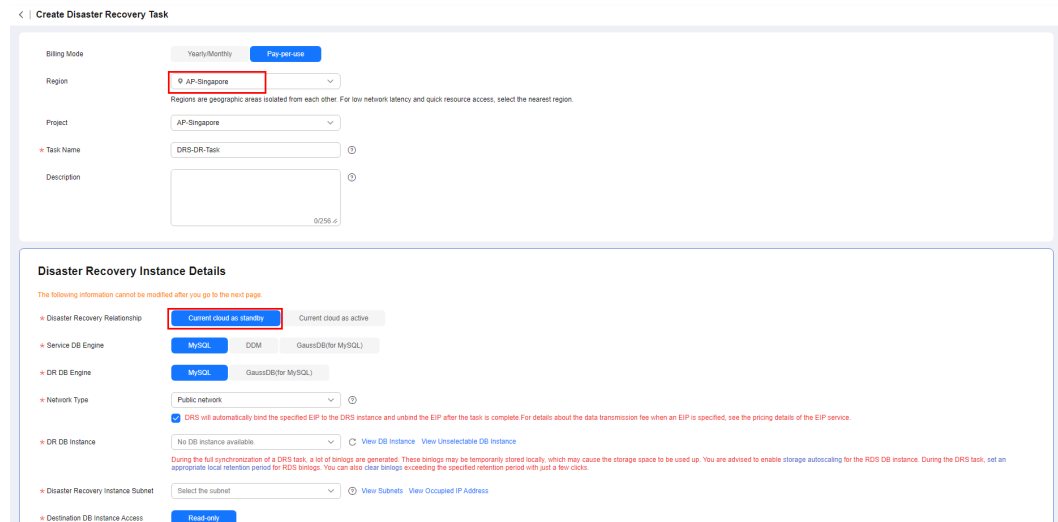
Create a DRS disaster recovery task in the same region as the RDS for MySQL instance configured for the DR center.

Procedure

Step 1 Go to the [Create Disaster Recovery Task](#) page.

Step 2 Select **AP-Singapore** for **Region**. Set **Disaster Recovery Relationship** to **Current cloud as standby**, and **DR DB Instance** to the RDS for MySQL DR instance created in the AP-Singapore region, and click **Create Now**.

Figure 2-25 Setting DR instance information



Step 3 Return to the **Disaster Recovery Management** page and check the status of the task.

----End

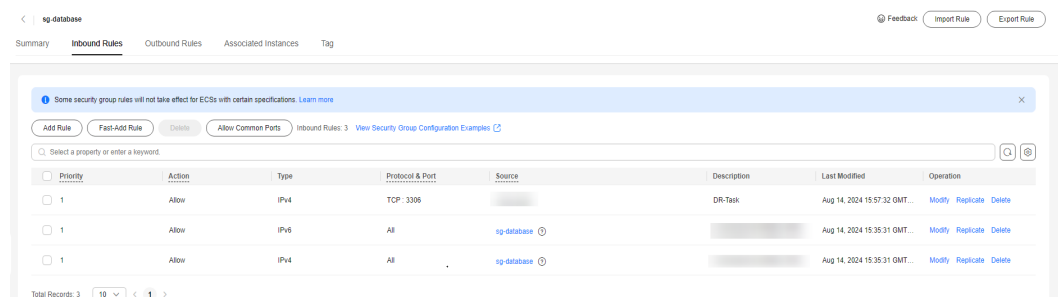
2.2.6.2 Configuring the Disaster Recovery Task

Configure the disaster recovery task, including setting the source and destination databases.

Procedure

- Step 1** On the **Disaster Recovery Management** page, locate the created disaster recovery task and click **Edit** in the **Operation** column.
- Step 2** Add the EIP of the DRS instance to the inbound rule of the security group associated with the RDS for MySQL instance in the production center, select TCP, and set the port number to that of the RDS for MySQL instance of the production center.

Figure 2-26 Adding a security group rule



In the **Source Database** area, set **IP Address or Domain Name** and **Port** to the EIP and port of the RDS for MySQL instance in the production center. When the connection test is successful, click **Next**.

Figure 2-27 Editing a disaster recovery task

Source Database

Database Type Self-built on ECS RDS DB instance

IP Address or Domain Name

Port

Database Username

Database Password

SSL Connection

Test Connection This button is available only after the replication instance is created successfully.

Step 3 Configure the flow control and click **Next**.

Figure 2-28 Configuring flow control

Flow Control Yes No ?

Migrate Definer to User Yes ? No ?

Step 4 Check the disaster recovery task. When the check success rate reaches 100%, click **Next**.

Step 5 Configure parameters and click **Next**.

Figure 2-29 Configuring parameters

Parameter Type Common parameters Performance parameters

Select the destination database parameters whose values you want to change to be the same as those in the source database. Some changes take effect only after you restart the destination database. You are advised to restart the destination database before or after the migration.

Save Change C

Parameter Name	Source Database Value	Destination Database Value	Result
<input type="checkbox"/> <input type="radio"/> connect_timeout	10	10	● Consistent
<input type="checkbox"/> <input type="radio"/> explicit_defaults_for_timestamp	OFF	OFF	● Consistent
<input type="checkbox"/> <input type="radio"/> innodb_flush_log_at_trx_commit	1	1	● Consistent
<input type="checkbox"/> <input type="radio"/> innodb_lock_wait_timeout	50	50	● Consistent
<input type="checkbox"/> <input type="radio"/> max_connections	6000	2500	● Inconsistent
<input type="checkbox"/> <input type="radio"/> net_read_timeout	30	30	● Consistent
<input type="checkbox"/> <input type="radio"/> net_write_timeout	60	60	● Consistent
<input type="checkbox"/> <input type="radio"/> transaction_isolation	REPEATABLE-READ	REPEATABLE-READ	● Consistent

Step 6 Configure **Start Time** and click **Submit**.

Figure 2-30 Starting the task

The screenshot shows a configuration panel for starting a task. It includes the following elements:

- Start Time:** Two radio buttons. The first, "Start upon task creation", is selected and highlighted in blue. The second is "Start at a specified time".
- Send Notifications:** A toggle switch that is turned on (blue).
- SMN Topic:** A dropdown menu with a downward arrow, a refresh icon, and a help icon.
- Delay Threshold (s):** A toggle switch that is turned off (grey).
- RTO Delay Threshold (s):** A toggle switch that is turned off (grey).
- RPO Delay Threshold (s):** A toggle switch that is turned off (grey).
- Stop Abnormal Tasks After:** A text input field containing the number "14", followed by a help icon and a red note: "Abnormal tasks run longer than the period you set (unit: day) will automatically stop."

Step 7 On the **Disaster Recovery Management** page, check the task status. The status is **Disaster recovery in progress**.

For a task that is in the **Disaster recovery in progress** state, you can use **data comparison** to check whether data is consistent before and after the disaster recovery.

----End

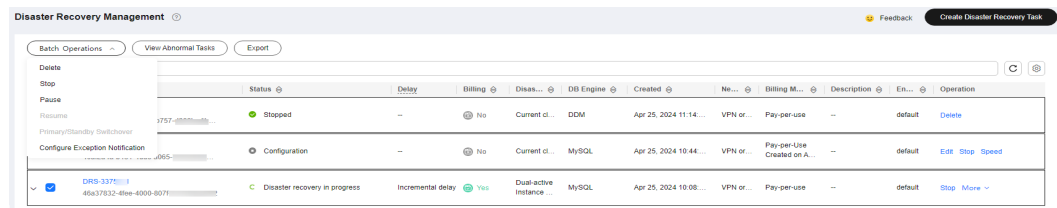
2.2.6.3 Performing a Primary/Standby Switchover

If the source database in the production center is faulty, manually switch the DR instance to the read/write state. Then, data is written to the DR instance and synchronized to the source database.

Procedure

- Step 1** Find that the source database in the production center is faulty. For example, the source database cannot be connected, the source database execution is slow, or the CPU usage is high.
- Step 2** Receive an SMN email notification.
- Step 3** Check the delay of the DR task.
- Step 4** Check that the services of the source database have been stopped. For details, see [How Do I Ensure that All Services on the Database Are Stopped?](#)
- Step 5** Select the task, click the **Batch Operations** drop-down box in the upper left corner and select **Primary/Standby Switchover**.

Figure 2-31 Primary/standby switchover



Step 6 Change the database IP address on your application and use it to connect to the database. Then data is properly read from and written to the database.

----End

2.3 Migrating MySQL Databases from Other Clouds to RDS for MySQL

2.3.1 Overview

Scenarios

This best practice includes the following tasks:

- Create an RDS for MySQL instance.
- Migrate data from a MySQL database on other clouds to RDS for MySQL.

Prerequisites

- You have registered with Huawei Cloud.
- Your account balance is at least \$0 USD.

Service List

- Virtual Private Cloud (VPC)
- RDS
- Data Replication Service (DRS)

Before You Start

- The resource planning in this best practice is for demonstration only. Adjust it as needed.
- All settings in this best practice are for reference only. For more information about MySQL migration, see [From MySQL to MySQL](#).

2.3.2 Resource Planning

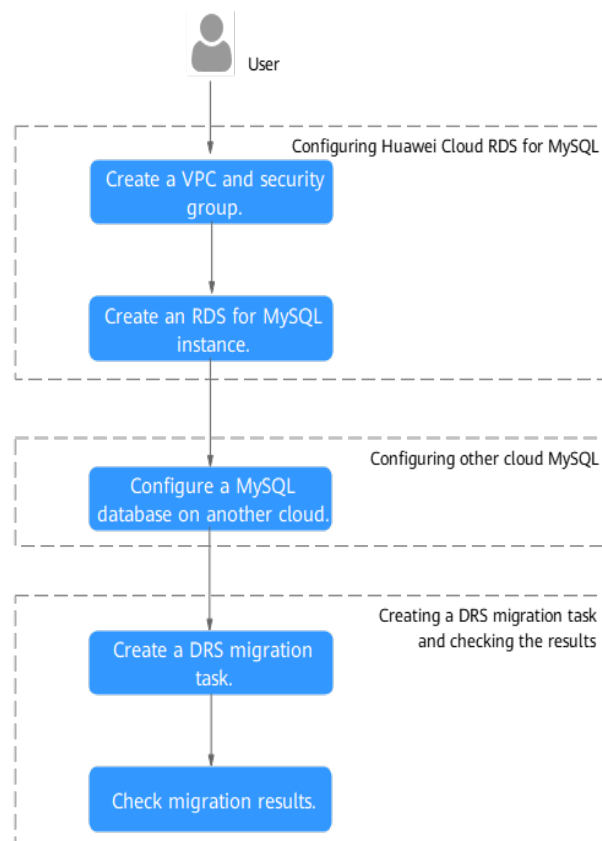
Table 2-3 Resource planning

Category	Subcategory	Planned Value	Description
VPC	VPC name	vpc-src-172	Specify a name that is easy to identify.
	Region	Test region	To achieve lower network latency, select the region nearest to you.
	AZ	AZ3	-
	Subnet	172.16.0.0/16	Select a subnet with sufficient network resources.
	Subnet name	subnet-src-172	Specify a name that is easy to identify.
MySQL on another cloud	Database version	MySQL 5.7	-
	IP address	10.154.217.42	Enter an IP address.
	Port	3306	-
RDS for MySQL instance	Instance name	rds-mysql	Specify a name that is easy to identify.
	DB engine version	MySQL 5.7	-
	Instance type	Single	A single instance is used in this example. To improve service reliability, select a primary/standby instance.
	Storage type	Cloud SSD	-
	AZ	AZ1	AZ1 is selected in this example. To improve service reliability, select the primary/standby instance type and deploy the primary and standby instances in different AZs.
	Instance class	General-purpose 2 vCPUs 8 GB	-
DRS migration task	Task name	DRS-mysql	Specify a name that is easy to identify.

Category	Subcategory	Planned Value	Description
	Source DB engine	MySQL	-
	Destination DB engine	MySQL	-
	Network type	Public network	Public network is used in this example.

2.3.3 Operation Process

Figure 2-32 Flowchart



2.3.4 Creating a VPC and Security Group

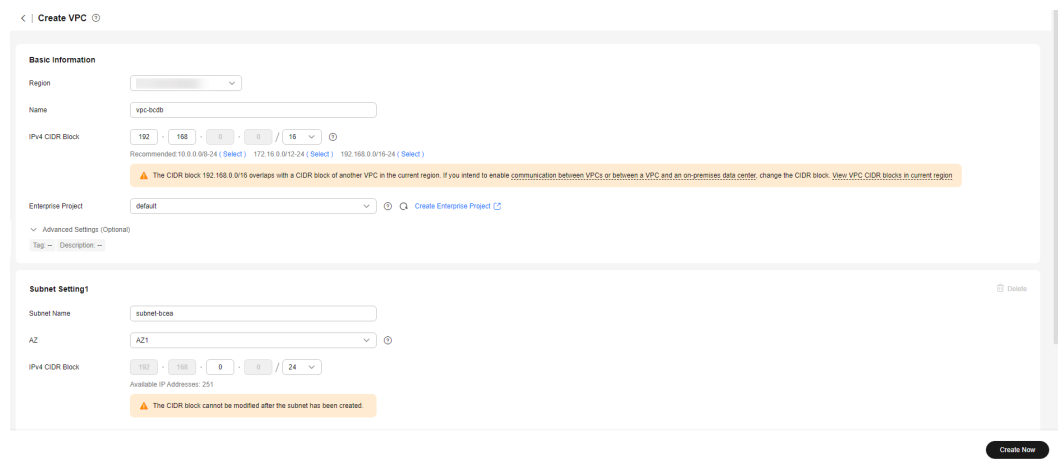
Create a VPC and security group for an RDS for MySQL instance

Creating a VPC

Step 1 Go to the [Create VPC](#) page.

Step 2 Configure the basic information, subnet, and IP address.

Figure 2-33 Creating a VPC



Step 3 Click **Create Now**.


Step 4 Return to the VPC list and check whether the VPC is created.

If the VPC status becomes available, the VPC has been created.

----End

Creating a Security Group

Step 1 Log in to the [management console](#).

Step 2 Click  in the upper left corner of the management console and select **CN-Hong Kong**.

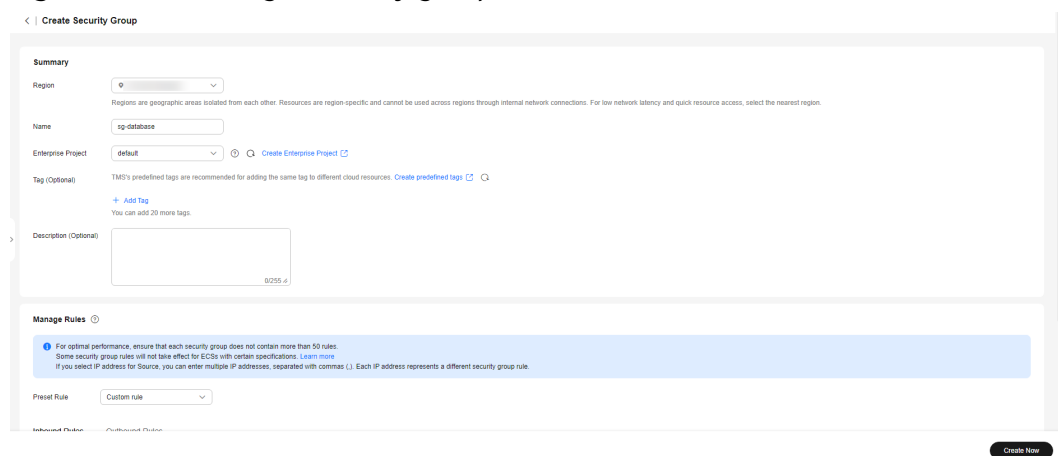
Step 3 Under the service list, choose **Networking > Virtual Private Cloud**.

Step 4 In the navigation pane, choose **Access Control > Security Groups**.

Step 5 Click **Create Security Group**.

Step 6 Configure parameters as needed.

Figure 2-34 Creating a security group



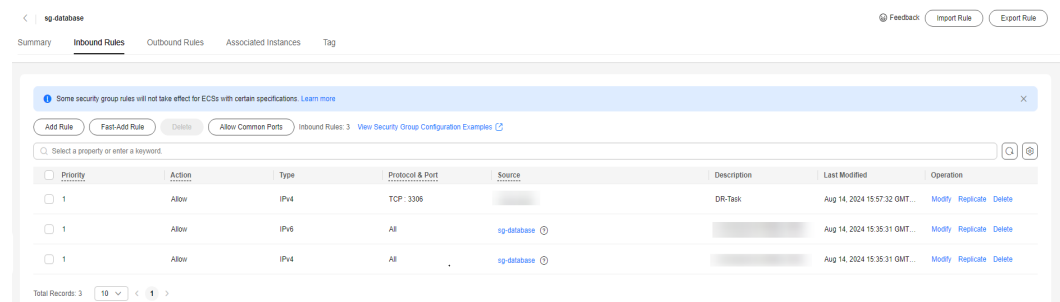
Step 7 Click **Create Now**.

Step 8 Return to the security group list and click the security group name.

Step 9 Click the **Inbound Rules** tab, and then click **Add Rule**.

Step 10 Configure an inbound rule to allow access from database port **3306**.

Figure 2-35 Inbound rules



----End

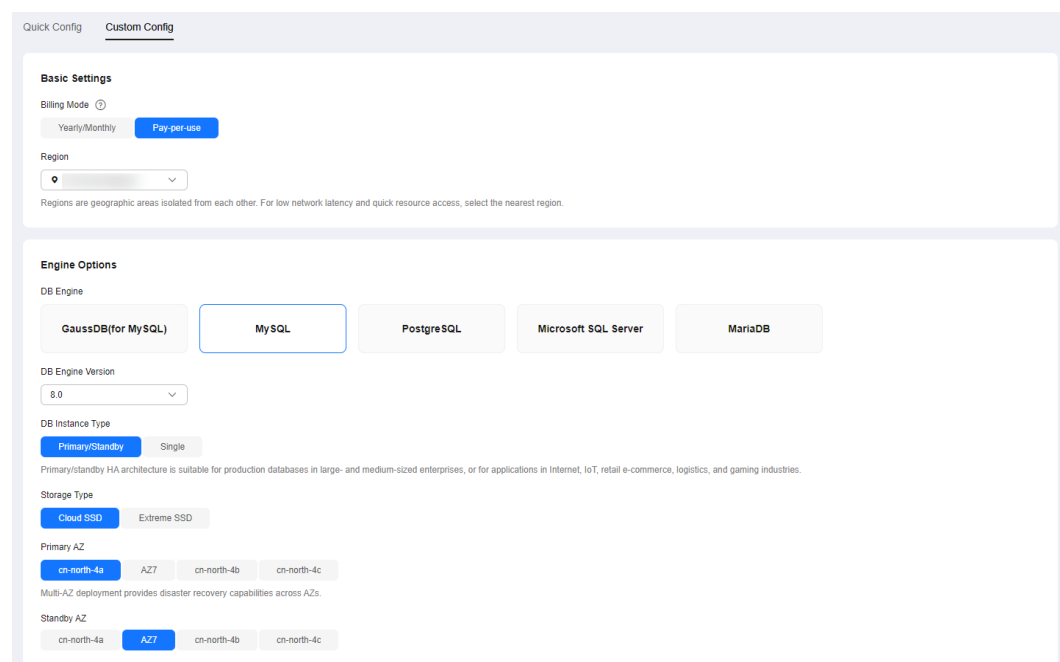
2.3.5 Creating an RDS for MySQL Instance

Create an RDS for MySQL instance, and select the VPC and security group you configured for the instance.

Step 1 Go to the **Buy DB Instance** page.

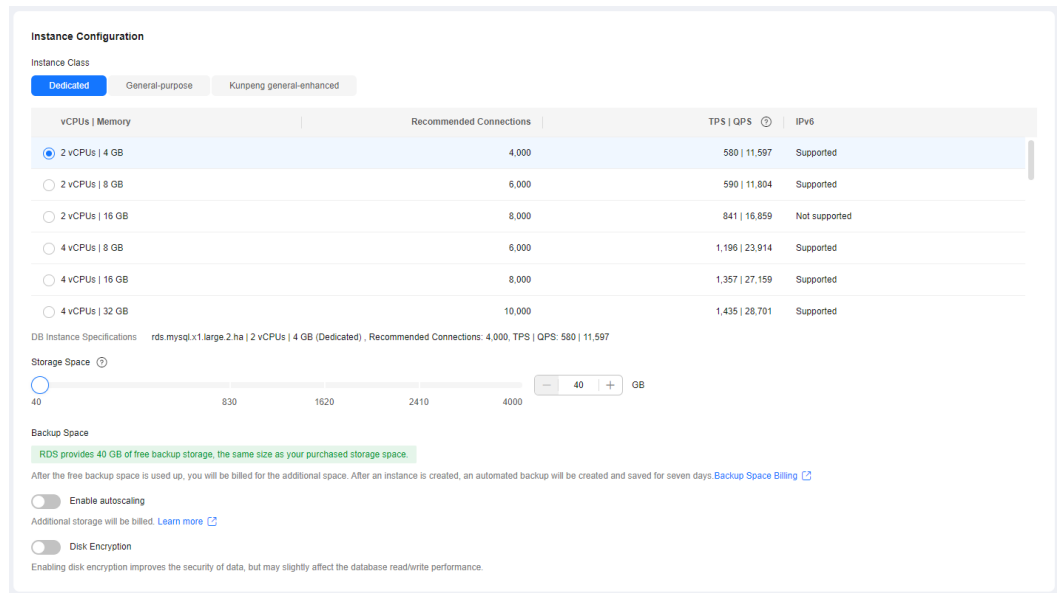
Step 2 Configure basic information for the instance. Select **CN-Hong Kong** for **Region**.

Figure 2-36 Basic information



Step 3 Select an instance class.

Figure 2-37 Instance class



Step 4 Select a VPC and security group for the instance and configure the database port. The VPC and security group have been created in [Creating a VPC and Security Group](#).

Figure 2-38 Network configurations

The screenshot displays the 'Basic Settings' and 'Connectivity' sections of the RDS console. In the 'Basic Settings' section, the 'DB Instance Name' is set to 'rds-9812'. Below this, there is a 'Password' section with 'Skip' and 'Configure' buttons. A note indicates that if multiple instances are bought, they will be named with four digits appended in the format 'DB instance name-SN'. In the 'Connectivity' section, the 'VPC' is set to 'vpc-a33b' with a 'Create VPC' link. The 'Subnet' is set to 'subnet-a389/192.168.0.0/24' with a search icon. Below the subnet selection, the IPv6 CIDR block is shown as '2407:c080:1200:217e::/64' and a note states that an EIP is required for public network access. The 'IPv4 Address' field is empty, with 'Addresses available: 251' and a 'View In-use IP Addresses' link. The 'Database Port' is set to 'Default port: 3306'. The 'Security Group' is set to 'sg-database' with a 'View Security Group' link. A note at the bottom of the connectivity section states that port 3306 of the security group allows traffic from the server IP address to the DB instance, with a 'Create Security Group' link.

Step 5 Complete advanced settings.

Figure 2-39 Additional options

Additional Options

Enterprise Project ⓘ
default Q [Create Enterprise Project](#)

Parameter Template
Default-HighPerformance-Edition-MySQL-8.0 Q [View Parameter Template](#)

⚠ Using a high-performance template tends to result in lost data and replication exceptions after an instance recovers from a crash. There may also be out of memory (OOM) errors with small instance classes in high concurrency scenarios. For details, see the user guide. [View Details](#)

Time Zone
(UTC+08:00) Beijing, Chongqing, Hong Kong, Urumqi

Table Name ⓘ
Case sensitive **Case insensitive**

Tag
TMS's predefined tags are recommended for adding the same tag to different cloud resources. [Create predefined tags](#) Q
[+ Add Tag](#)
You can add 20 more tags.

Read Replica
Skip Create

Required Duration and Quantity
Quantity
- 1 +
You can create 50 more instances (read replicas included). [Increase Quota](#)

Step 6 Click **Next**.

Step 7 Confirm the settings.

- To modify your settings, click **Previous**.
- If you do not need to modify your settings, click **Submit**.

Step 8 Return to the instance list. If the instance status becomes available, the instance has been created.

----End

2.3.6 Configuring a MySQL Instance on Another Cloud

Prerequisites

- You have purchased a MySQL instance from another cloud vendor platform.
- Your account has the migration permissions listed in [Permission Requirements](#).

Permission Requirements

Table 2-4 lists the permissions required for migrating data from a MySQL instance on another cloud to RDS for MySQL using DRS. For details about the permissions, see [Which MySQL Permissions Are Required for DRS?](#)

Table 2-4 Migration permissions

Database	Full Migration Permission	Full+Incremental Migration Permission
Source database (MySQL)	SELECT, SHOW VIEW, and EVENT	SELECT, SHOW VIEW, EVENT, LOCK TABLES, REPLICATION SLAVE, and REPLICATION CLIENT

Network Configuration

You need to enable public accessibility for the source database.

Whitelist Settings

The EIP of the DRS replication instance must be on the whitelist of the source database for the connectivity between the DRS replication instance and the source database. To obtain the EIP of the DRS replication instance, see [Step 3 in Creating a DRS Migration Task](#). This method of configuring a whitelist varies depending on the cloud database vendors. For details, see their official documents.

2.3.7 Cloud Migration

2.3.7.1 Creating a DRS Migration Task

Creating a Migration Task

Step 1 Go to the [Create Migration Task](#) page.

Step 2 Configure parameters as needed.

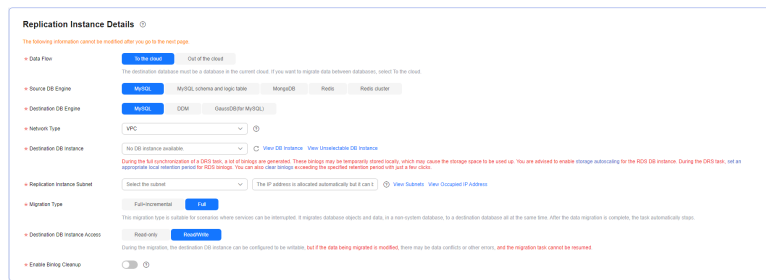
1. Enter the migration task name. Select the region hosting the destination DB instance for **Region**.

Figure 2-40 Migration task

2. Configure the replication instance information.

Select the RDS instance created in [Creating an RDS for MySQL Instance](#) as the destination database.

Figure 2-41 Replication instance details



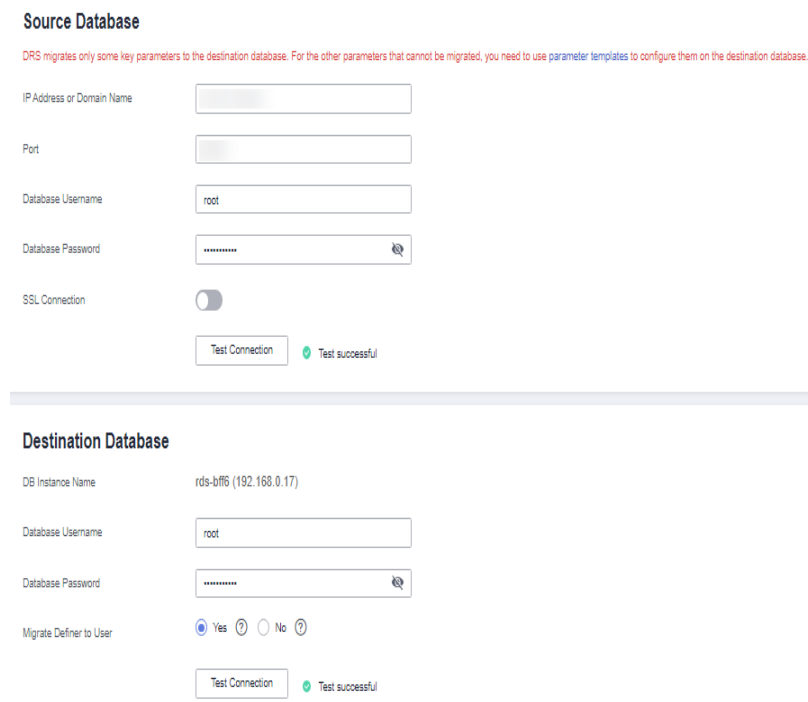
Step 3 Click **Create Now**.

It takes about 5 to 10 minutes to create a replication instance. After the replication instance is created, you can obtain its EIP.

✔ The replication instance is created. Its EIP is 122.9.214.142. Add this EIP to the source database whitelist so that it can access the source database.

Step 4 Configure the source and destination database information.

Figure 2-42 Configuring the source and destination databases



Step 5 Click **Next**.

Step 6 On the **Set Task** page, configure parameters as required.

- Set **Flow Control** to **No**.
- Set **Migration Object** to **All**.

Step 7 Click **Next**. On the **Check Task** page, check the migration task.

- If any check fails, review the cause and rectify the fault. After the fault is rectified, click **Check Again**.
- If all check items are successful, click **Next**.

Step 8 Compare source and destination database parameters.

- If you do not want to compare the parameters, click **Next** to skip this step.
- If there are inconsistent common parameter values, click **Save Change** to change the destination database values to match those of the source database.

Step 9 Click **Submit** to submit the task.

Return to the **Online Migration Management** page and check the migration task status.

It takes several minutes to complete.

If the status changes to **Completed**, the migration task is complete.

----End

2.3.7.2 Checking Migration Results

You can use either of the following methods to check the migration results:

1. Use DRS to compare migration objects, users, and data of source and destination databases and obtain the migration results. For details, see [Checking the Migration Results on the DRS Console](#).
2. Log in to the destination instance to check whether the databases, tables, and data are migrated. For details, see [Checking the Migration Results on the RDS Console](#).

Checking the Migration Results on the DRS Console

Step 1 Log in to the [management console](#).

Step 2 Click  in the upper left corner and select your region.

Step 3 Under the service list, choose **Databases > Data Replication Service**.

Step 4 Click the DRS instance name.


Step 5 Click **Migration Comparison** in the navigation pane. Under the **Object-Level Comparison** tab, click **Compare** to check whether all objects have been migrated to the destination instance.

Step 6 Click the **Data-Level Comparison** tab. On the displayed page, click **Create Comparison Task** to check whether the databases and tables of the source and destination instances are the same.

Step 7 Click **Account-Level Comparison** and check whether the accounts and permissions of the source and destination instances are the same.

----End

Checking the Migration Results on the RDS Console

- Step 1** Log in to the [management console](#).
- Step 2** Click  in the upper left corner and select your region.
- Step 3** Click the service list icon on the left and choose **Databases > Relational Database Service**.
- Step 4** Locate the destination instance and click **Log In** in the **Operation** column.
- Step 5** In the displayed dialog box, enter the password and click **Test Connection**.
- Step 6** After the connection test is successful, click **Log In**.
- Step 7** Check whether the databases and tables of the source instance have been migrated.

----End

Performing a Performance Test

After the migration is complete, you can perform a performance test as required.

2.4 Using RDS for MySQL to Set Up WordPress

WordPress is a blog platform developed based on PHP. It is usually used with RDS for MySQL database servers to help users build websites. This section describes how to set up WordPress in the Linux, Apache, MySQL and PHP (LAMP) environment using Huawei Cloud VPC, ECS, and RDS for MySQL.

1. [Configuring Network Information](#)
2. [Buying an ECS](#)
3. [Setting Up the LAMP Environment](#)
4. [Buying and Configuring an RDS DB Instance](#)
5. [Installing WordPress](#)

Preparations

During the setup, you will use the following services or tools:


- Cloud services: Huawei Cloud ECS and RDS for MySQL.
- MySQL client: a database configuration tool
- PuTTY: a remote login tool

NOTE

The previous software is provided by third-party websites. The information is just for your reference and not for commercial use.

Configuring Network Information

- Step 1** Log in to the [management console](#).

- Step 2** Click  in the upper left corner and select a region.
 - Step 3** Choose **Networking > Virtual Private Cloud**.
 - Step 4** On the displayed page, click **Create VPC** to create a VPC, such as vpc-01.
 - Step 5** On the displayed page, enter a VPC name, set **IPv4 CIDR Block** to **192.168**, select an AZ as required, and add a subnet. Retain the default settings for other parameters. Then, click **Create Now**. After the VPC is created, return to the network console.
 - Step 6** On the network console, choose **Access Control > Security Groups** and click **Create Security Group**. The following uses sg-01 as an example.
 - Step 7** On the **Security Groups** page, locate the target security group and click **Manage Rules** in the **Operation** column.
 - Step 8** Click **Add Rule** and add an inbound rule for the **EIP** bound to the ECS.
- End

Buying an ECS


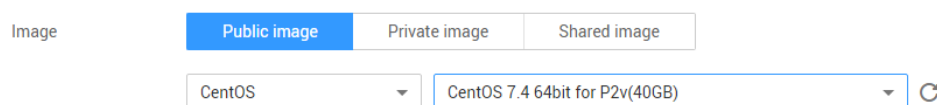
- Step 1** Log in to the [management console](#).
- Step 2** Click  in the upper left corner and select a region.
- Step 3** Choose **Compute > Elastic Cloud Server**. The **Elastic Cloud Server** page is displayed.
- Step 4** On the ECS console, buy an ECS.
 1. Configure basic settings: Select the pay-per-use billing mode, a region, and an image. Retain the default settings for other parameters.
The public image **CentOS7.4 64bit for P2v(40GB)** is used as an example, as shown in [Figure 2-43](#).

Figure 2-43 Selecting an image



2. Configure network: Select a VPC and security group, and purchase an EIP. Retain the default settings for other parameters.
 - a. Select the created VPC vpc-01.
 - b. Select the created security group sg-01.
 - c. Select **Auto assign** for **EIP**.
3. Configure advanced settings: Enter an ECS name and password, and click **Next: Confirm**.
 - a. Enter an ECS name, such as *ecs-01*.
 - b. Enter a password.
4. Confirm: Confirm the information and click **Next**.

Step 5 After the ECS is created, view and manage it on the ECS console.

----End

Setting Up the LAMP Environment

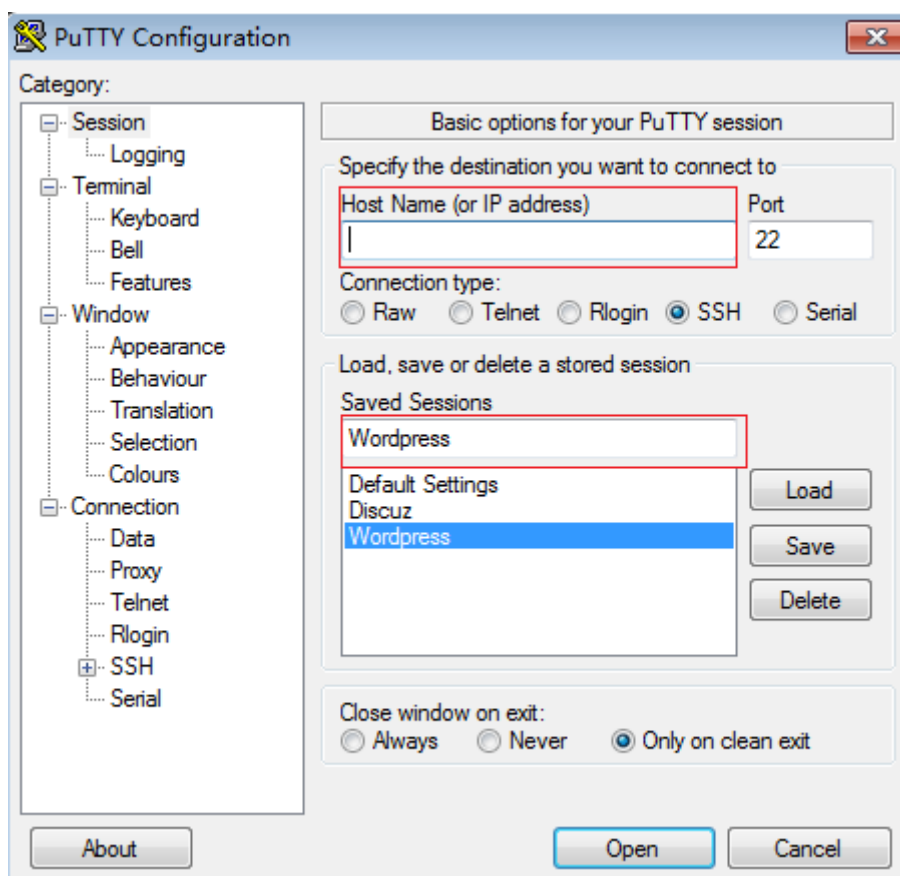
Step 1 Download the PuTTY client.

Step 2 Decompress the package, locate **putty** from the extracted files and double-click it.

Step 3 In the displayed PuTTY configuration dialog box, choose **Session** and specify basic options for your PuTTY session in the right pane. Then, click **Open** as shown in [Figure 2-44](#).

1. Enter the EIP of your ECS in the **Host Name (or IP address)** text box.
2. Enter a session name in the **Saved Sessions** text box and click **Save**. **Wordpress** is used as an example. Retain the default settings for other parameters.

Figure 2-44 Configuring PuTTY



Step 4 In the displayed login window, enter the ECS username and password to log in to ECS.

Step 5 Obtain the **root** permissions so that you can enter commands in PuTTY.

Enter commands to install MySQL, PHP or other software. For example, run the following command to install PHP:

```
yum install -y httpd php php-fpm php-server php-mysql mysql
```

The installation is complete if the following command output is displayed:
Complete

Step 6 Run the following command to install a decompression software:

```
yum install -y unzip
```

Step 7 Run the following command to download and decompress the WordPress installation package:

```
wget -c https://wordpress.org/wordpress-4.9.1.tar.gz
```

```
tar xzf wordpress-4.9.1.tar.gz -C /var/www/html
```

```
chmod -R 777 /var/www/html
```

Step 8 After the installation is complete, run the following commands to start related services in sequence:

```
systemctl start httpd.service
```

```
systemctl start php-fpm.service
```

Step 9 Enable automatic start of the service during system startup.

```
systemctl enable httpd.service
```

----End

Buying and Configuring an RDS DB Instance

Step 1 [Buy a DB instance](#) as required.

- DB instance rds-01 is used as an example. Select MySQL 5.7.
- Ensure that the RDS DB instance uses the same security group as the ECS so that you can access the RDS DB instance through the ECS.
- Set the root user password and keep the password secure. The system cannot retrieve your password.

Step 2 Go to the RDS console. On the **Instances** page, click the target DB instance rds-01. The **Overview** page is displayed.

Step 3 Choose **Databases** in the navigation pane on the left and click **Create Database**. In the displayed dialog box, enter a database name, such as **wordpress**, select a character set, and authorize permissions for database users. Then, click **OK**.

Figure 2-45 Creating a database

Create Database ×

Database Name ⓘ

Character Set utf8 gbk latin1 utf8mb4 [More](#)

User

User Not Authorized 0 / 0

🔍 Enter a keyword.

<input type="checkbox"/> Name	Permission
No data available	

Authorized User 0 / 0

🔍 Enter a keyword.

<input type="checkbox"/> Name	Permission
No data available	

Remarks 0/512

If you require fined-grained authorization, [log in to the database](#).

Step 4 Choose **Accounts** in the navigation pane on the left and click **Create Account**. In the displayed dialog box, enter the database username, such as **tony**, authorize permissions for database **wordpress** created in **Step 3**, enter the password and confirm the password. Then, click **OK**.

Figure 2-46 Creating an account

Create Account ✕

Username ⓘ

Host IP Address ⓘ

Database

Database Not Authorized 0 / 0

🔍 Enter a keyword.

<input type="checkbox"/> Name	Permission
No data available	

Database Authorized 0 / 1

🔍 Enter a keyword.

<input type="checkbox"/> Name	Permission
<input type="checkbox"/> wordpress	<input type="radio"/> Read only <input checked="" type="radio"/> Read and write

Password ⓘ

Confirm Password ⓘ

If you require fined-grained authorization, [log in to the database.](#)

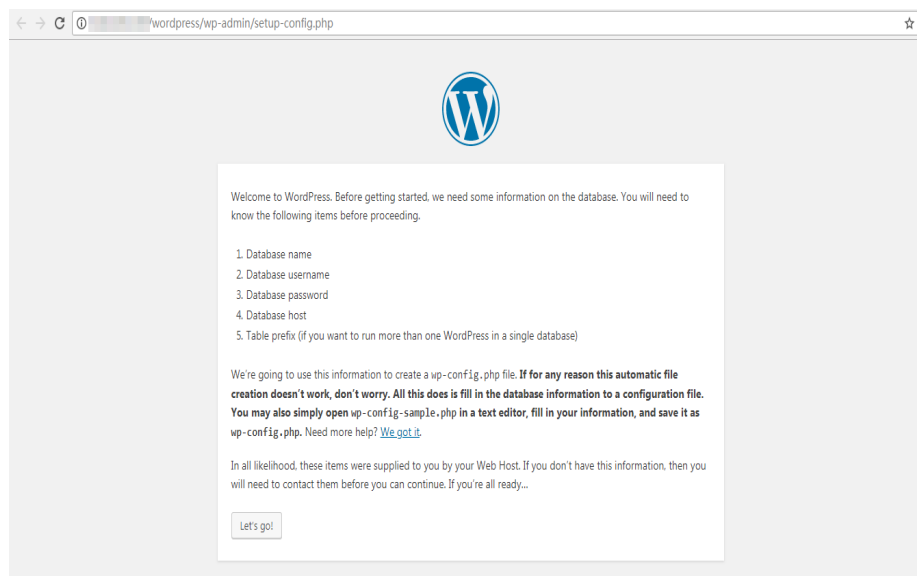
----End

Installing WordPress

- Step 1** On the **Elastic Cloud Server** page, locate the target ECS and click **Remote Login** in the **Operation** column.
- Step 2** In the Internet Explorer, enter **http://EIP/wordpress** in the address box and click **Let's go!**

In the preceding URL, **EIP** indicates the EIP automatically assigned when you purchase the ECS in **Buying an ECS**.

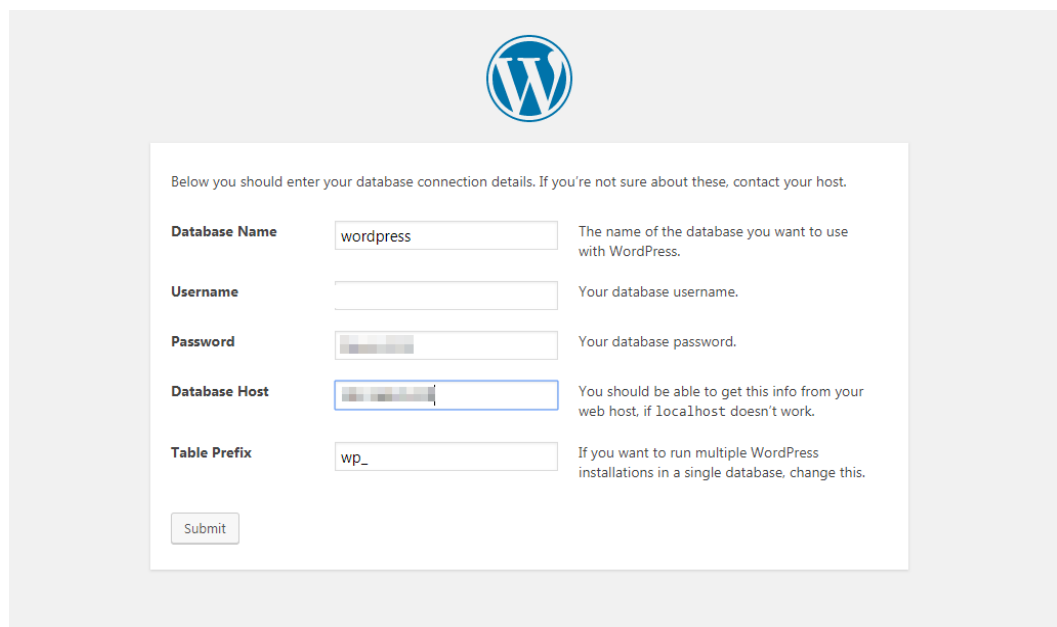
Figure 2-47 Visiting WordPress



Step 3 Enter database connection information and click **Submit**.

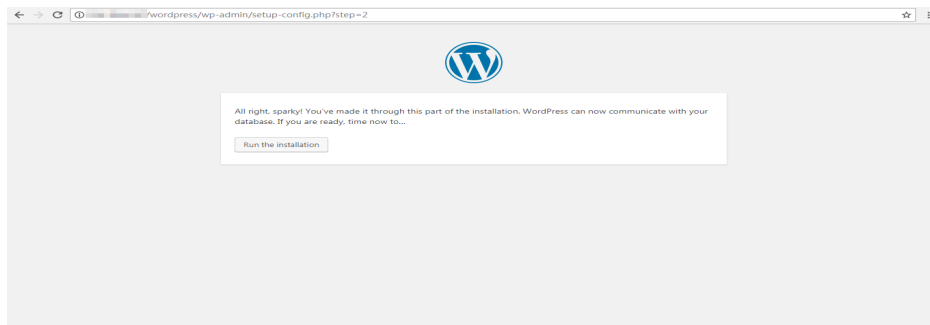
- The database name is *wordpress*.
- The username is *tony*.
- The password is the one that you set for *tony*.
- The database host is the floating IP address of DB instance rds-01.

Figure 2-48 Entering database connection information



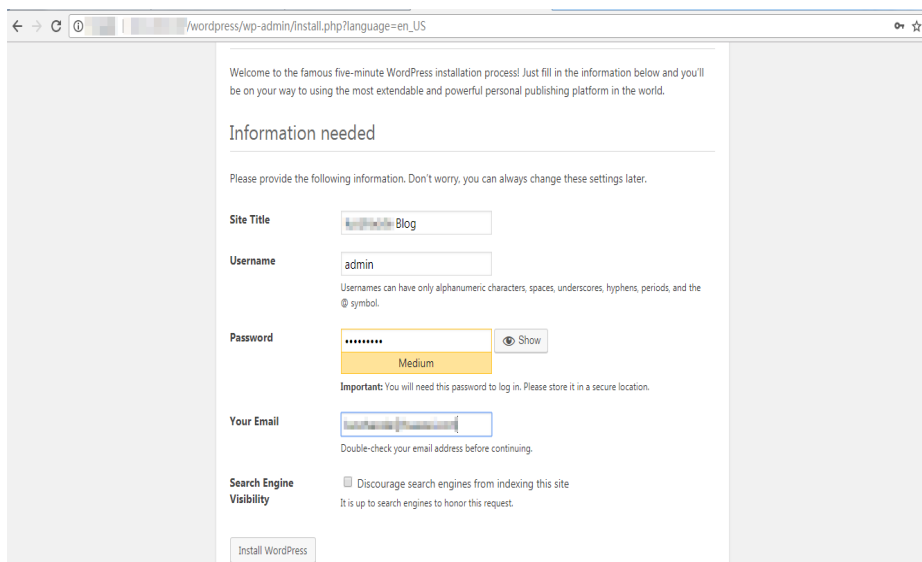
Step 4 After the database connection details are verified, click **Run the installation**.

Figure 2-49 Running the installation



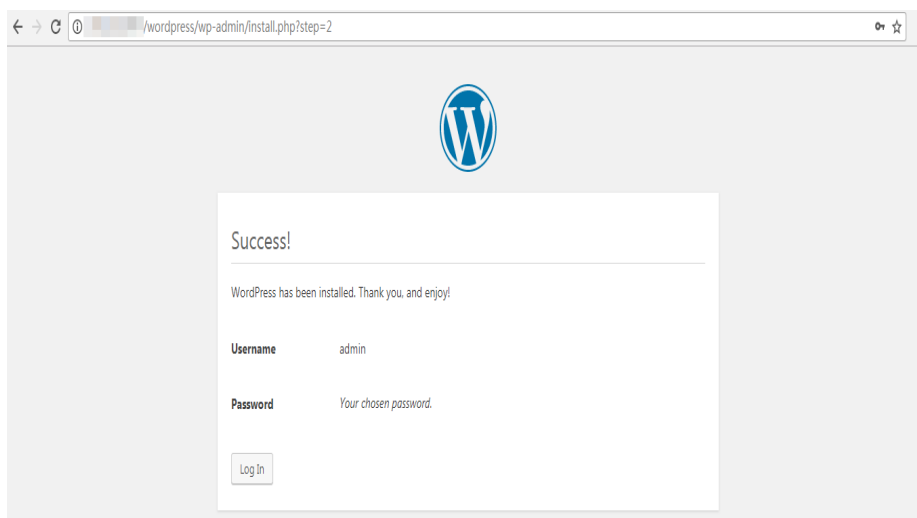
Step 5 Set **Site Title**, **Username**, and **Password** for logging in to your blog. Then, click **Install WordPress**.

Figure 2-50 Setting basic information



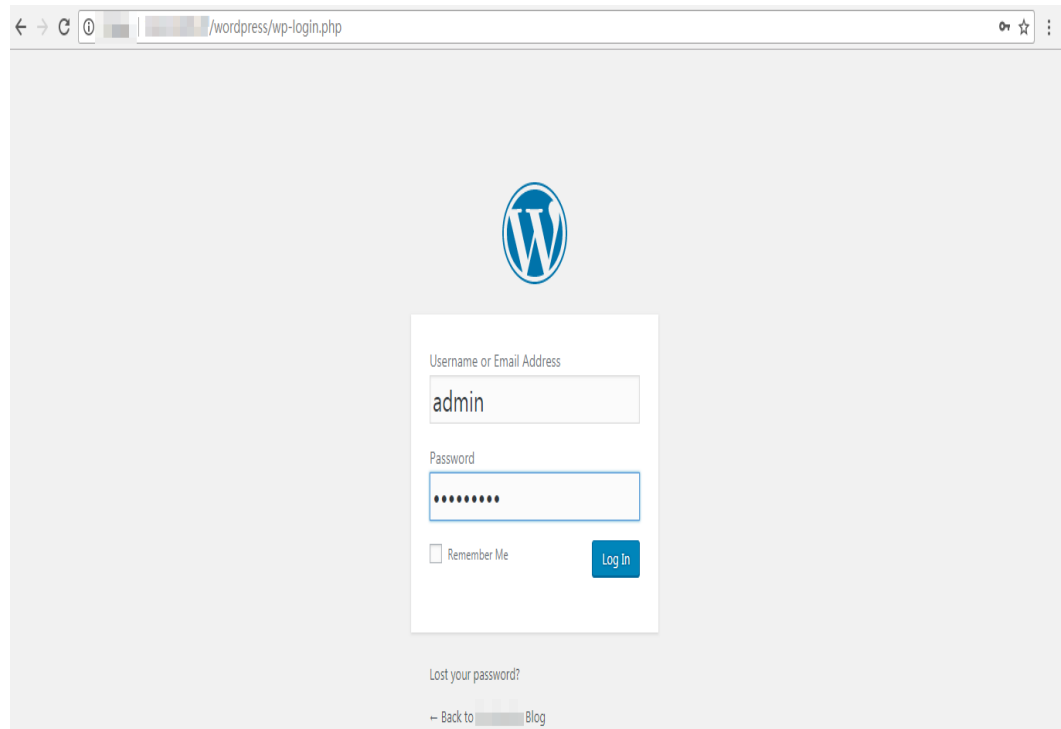
Step 6 Click **Log In** after WordPress has been successfully installed.

Figure 2-51 Successful installation



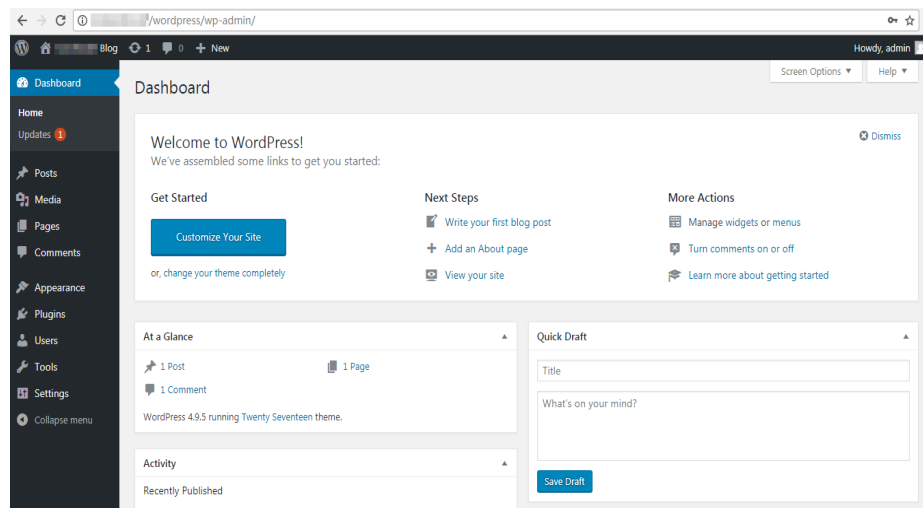
Step 7 Enter the username and password on the displayed login page. Then, click **Log In**.

Figure 2-52 Logging in



Step 8 Check that WordPress has been deployed successfully.

Figure 2-53 Verification



----End

2.5 Using RDS for MySQL to Set Up Discuz!

Crossday Discuz! Board (Discuz! for short) is a universal community forum software system. You can set up a customized forum with comprehensive functions and strong load capability on the Internet through simple installation

and settings. This section describes how to set up Discuz! in the LAMP environment using Huawei Cloud VPC, ECS, and RDS for MySQL.

1. [Configuring Network Information](#)
2. [Creating an ECS](#)
3. [Setting Up the LAMP Environment](#)
4. [Buying and Configuring an RDS DB Instance](#)
5. [Installing Discuz!](#)

Preparations

During the setup, you will use the following services or tools:


- Cloud services: ECS and RDS on Huawei Cloud
- PuTTY: a remote login tool
- Installation packages
 - Apache Http Server 2.4.6
 - MySQL 5.4.16
 - PHP 5.4.16

NOTE

The previous software is provided by third-party websites. The information is just for your reference and not for commercial use.

Configuring Network Information

Step 1 Log in to the [management console](#).

Step 2 Click  in the upper left corner and select a region.

Step 3 Choose **Networking > Virtual Private Cloud**.

Step 4 On the displayed page, click **Create VPC** to create a VPC, such as vpc-01.

Step 5 On the displayed page, enter a VPC name, set **IPv4 CIDR Block** to **192.168**, select an AZ as required, and add a subnet. Retain the default settings for other parameters. Then, click **Create Now**. After the VPC is created, return to the network console.

Step 6 On the network console, choose **Access Control > Security Groups** and click **Create Security Group**. The following uses sg-01 as an example.


Step 7 On the **Security Groups** page, locate the target security group and click **Manage Rules** in the **Operation** column.

Step 8 Click **Add Rule** and add an inbound rule for the **EIP** bound to the ECS.

----End

Buying an ECS

Step 1 Log in to the [management console](#).

Step 2 Click  in the upper left corner and select a region.

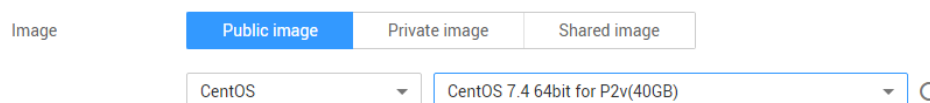
Step 3 Choose **Compute > Elastic Cloud Server**. The **Elastic Cloud Server** page is displayed.

Step 4 On the ECS console, buy an ECS.

1. Configure basic settings: Select the pay-per-use billing mode, a region, and an image. Retain the default settings for other parameters.

The public image **CentOS7.4 64bit for P2v(40GB)** is used as an example, as shown in [Figure 2-54](#).

Figure 2-54 Selecting an image



2. Configure network: Select a VPC and security group, and purchase an EIP. Retain the default settings for other parameters.
 - a. Select the created VPC vpc-01.
 - b. Select the created security group sg-01.
 - c. Select **Auto assign** for **EIP**.
3. Configure advanced settings: Enter an ECS name and password, and click **Next: Confirm**.
 - a. Enter an ECS name, such as *ecs-01*.
 - b. Enter a password.
4. Confirm: Confirm the information and click **Next**.

Step 5 After the ECS is created, view and manage it on the ECS console.

----End

Setting Up the LAMP Environment

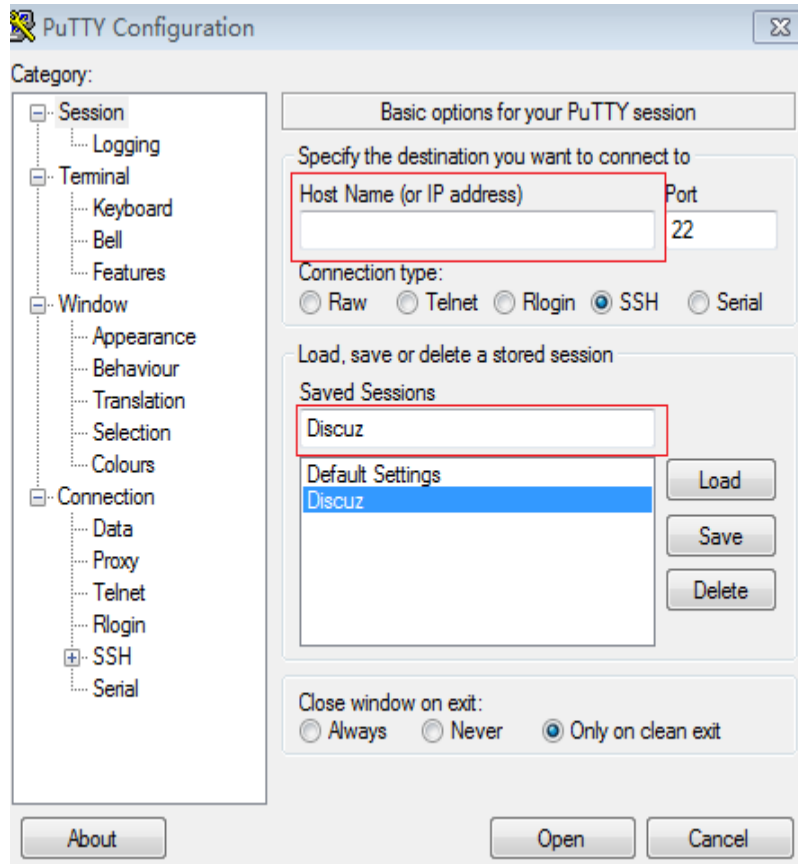
Step 1 Download the PuTTY client.

Step 2 Decompress the package, locate **putty** from the extracted files and double-click it.

Step 3 In the displayed PuTTY configuration dialog box, choose **Session** and specify basic options for your PuTTY session in the right pane. Then, click **Open** as shown in [Figure 2-55](#).

1. Enter the EIP of your ECS in the **Host Name (or IP address)** text box.
2. Enter a session name in the **Saved Sessions** text box and click **Save**. **Discuz** is used as an example. Retain the default settings for other parameters.

Figure 2-55 Configuring PuTTY



Step 4 In the displayed login window, enter the ECS username and password to log in to ECS.

Step 5 Install Apache, MySQL, PHP and other software.

Obtain the **root** permissions so that you can enter commands in PuTTY.

Enter commands to install software. For example, run the following command to install PHP:

```
yum install -y httpd php php-fpm php-server php-mysql mysql
```

The installation is complete if the following command output is displayed:
Complete

Step 6 After the installation is complete, start related services in sequence.

```
systemctl start httpd.service
```

```
systemctl start php-fpm.service
```

----End

Buying and Configuring an RDS DB Instance

Step 1 [Buy a DB instance](#) as required.

- DB instance rds-01 is used as an example. Select MySQL 5.7.

- Ensure that the RDS DB instance uses the same security group as the ECS so that you can access the RDS DB instance through the ECS.
- Set the root user password and keep the password secure. The system cannot retrieve your password.

Step 2 After the RDS DB instance is created, view or manage it on the [management console](#).

----End

Installing Discuz!

Step 1 Download the [Discuz! installation package](#).

Step 2 Upload the installation package to the ECS using a data transfer tool.

1. Run the following command to decompress the Discuz! installation package:
unzip Discuz_X3.3_SC_UTF8.zip
2. Run the following command to copy all files in **upload** to **/var/www/html/**.
cp -R upload/* /var/www/html/
3. Run the following command to grant write permissions to other users.
chmod -R 777 /var/www/html

Step 3 Enter **http://EIP/install** in the address box in a local Windows browser and install Discuz! following the guidance.

In the preceding URL, **EIP** indicates the EIP automatically assigned when you purchase the ECS in [Buying an ECS](#). The **install** must be lowercase.

1. Confirm the agreement and click **I Agree**.
2. After the installation starts, check the installation environment and click **Next**.
3. Set the running environment and click **Next**.
4. Enter the database information and click **Next** to complete the installation.
 - The database address is the floating IP address of DB instance rds-01.
 - The database password is the root user password of DB instance rds-01.
 - Enter administrator information.

Step 4 After Discuz! is installed, enter **http://EIP/forum.php** in the browser address bar. If the forum homepage is displayed, the website is successfully built.

----End

2.6 Description of `innodb_flush_log_at_trx_commit` and `sync_binlog`

The `innodb_flush_log_at_trx_commit` and `sync_binlog` are key parameters for controlling the disk write policy and data security of RDS for MySQL. Different parameter values have different impacts on performance and security.

Table 2-5 Parameter description

Parameter	Allowed Values	Description
innodb_flush_log_at_trx_commit	0, 1, and 2	Controls the balance between strict ACID compliance for commit operations, and higher performance that is possible when commit-related I/O operations are rearranged and done in batches. The default value is 1 . For details, see Parameter Description .
sync_binlog	0 to 4, 294, 967, 295	Sync binlog (RDS for MySQL flushes binary logs to disks or relies on the OS).

Parameter Description

- **innodb_flush_log_at_trx_commit:**
 - **0:** The log buffer is written out to the log file once per second and the flush to disk operation is performed on the log file, but nothing is done at a transaction commit.
 - **1:** The log buffer is written out to the log file at each transaction commit and the flush to disk operation is performed on the log file.
 - **2:** The log buffer is written out to the file at each commit, but the flush to disk operation is not performed on it. However, the flushing on the log file takes place once per second.

NOTE

- A value of **0** is the fastest choice but less secure. Any mysqld process crash can erase the last second of transactions.
 - A value of **1** is the safest choice because in the event of a crash you lose at most one statement or transaction from the binary log. However, it is also the slowest choice.
 - A value of **2** is faster and more secure than **0**. Only an operating system crash or a power outage can erase the last second of transactions.
- **sync_binlog=1 or N**
By default, the binary log is not every time synchronized to disk. In the event of a crash, the last statement in the binary log may get lost.
To prevent this issue, you can use the **sync_binlog** global variable (**1** is the safest value, but also the slowest) to synchronize the binary log to disk after N binary log commit groups.

Recommended Configurations

Table 2-6 Recommended configurations

<code>innodb_flush_log_at_trx_commit</code>	<code>sync_binlog</code>	Description
1	1	High data security and strong disk write capability
1	0	High data security and insufficient disk write capability. Standby lagging behind or no replication is allowed.
2	0/N (0 < N < 100)	Low data security. A small amount of transaction log loss and replication delay is allowed.
0	0	Limited disk write capability. No replication or long replication delay is allowed.

 **NOTE**

- When both `innodb_flush_log_at_trx_commit` and `sync_binlog` are set to 1, the security is the highest but the write performance is the lowest. In the event of a crash you lose at most one statement or transaction from the binary log. This is also the slowest choice due to the increased number of disk writes.
- When `sync_binlog` is set to N ($N > 1$) and `innodb_flush_log_at_trx_commit` is set to 2, the RDS for MySQL write operation achieves the optimal performance.

2.7 How Do I Improve the Query Speed of My RDS for MySQL Instance?

The following are some suggestions provided for you to improve the database query speed:

- View the slow query logs to check if there are any slow queries, and review their performance characteristics (if any) to locate the cause. For details about how to view RDS for MySQL logs, see [Viewing and Downloading Slow Query Logs](#).
- View the CPU usage of your RDS DB instance to facilitate troubleshooting. For details, see [Configuring Displayed Metrics](#).
- Create read replicas to offload read pressure on the primary DB instance. For details, see [Introducing Read Replicas](#).
- [Enable read/write splitting](#) after read replicas are created. Write requests are automatically routed to the primary DB instance and read requests are routed to read replicas by user-defined weights.
- Increase the CPU or memory specifications for DB instances with high load. For details, see [Changing a DB Instance Class](#). To temporarily reduce the load, you can kill sessions. For details, see [Managing Real-Time Sessions](#).

- Add indexes for associated fields in multi-table association queries.
- Specify a field or add a WHERE clause, which will prevent full table scanning triggered by the SELECT statement.

2.8 Handling RDS for MySQL Long Transactions

Potential Impacts of Long Transactions

1. Long transactions lock resources and usually increase metadata locks and row locks. As a result, other transactions cannot access these resources, reducing the database concurrency.
2. Long transactions may occupy a large amount of memory.
3. Long transactions may cause too large log files and high storage usage.

Identifying Long Transactions


- Connect to your DB instance and check long transactions and their session IDs.

After connecting to the DB instance, run the following command to view the ID of any transaction that has been executing for more than 3,000s, the executed SQL statement, and the corresponding session ID.

```
mysql> SELECT trx_id, trx_state, trx_started, trx_mysql_thread_id,
trx_query, trx_rows_modified FROM information_schema.innodb_trx
WHERE TIME_TO_SEC(timediff(now(),trx_started)) >3000;
```

Table 2-7 Parameter description

Parameter	Description
trx_id	Transaction ID.
trx_state	Transaction status, which can be RUNNING , LOCK WAIT , or ROLLING BACK .
trx_started	Time when the transaction was started.
trx_mysql_thread_id	ID of the MySQL session to which the transaction belongs.
trx_query	SQL statement executed by the transaction.
trx_rows_modified	Number of rows modified by the transaction.

- Check monitoring metrics for long transactions.
 - a. [Log in to the management console](#).
 - b. Click  in the upper left corner of the page and choose **Databases > Relational Database Service**.

- c. On the **Instances** page, locate the target DB instance and click **View Metrics** in the **Operation** column.
- d. Check the long transaction metric **rds_long_transaction**. If the metric increases linearly to a large value, there are long transactions.

Killing Long Transactions

1. Obtain the thread IDs corresponding to long transactions.
Run the SQL statement in [Connect to your DB instance to check long transactions and their session IDs](#) to obtain the session ID of the transaction whose execution time exceeds a certain period (for example, 3,000s).

```
mysql> SELECT trx_mysql_thread_id FROM
information_schema.innodb_trx WHERE
TIME_TO_SEC(timediff(now(),trx_started)) >3000;
```

2. After obtaining the session ID, run the **kill** command to kill the transaction.
`mysql> kill trx_mysql_thread_id`

NOTICE

Killing a long transaction will cause the transaction to roll back. Evaluate the impact before running this command.

Configuring Long Transaction Alarms


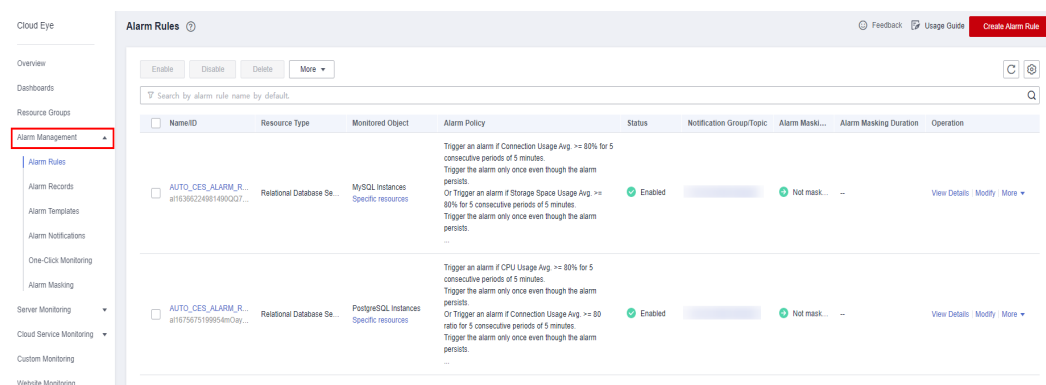

1. View the configured alarms.
 - a. [Log in to the management console](#).
 - b. Click  in the upper left corner of the page and choose **Management & Governance > Cloud Eye**.
 - c. Choose **Alarm Management > Alarm Rules**.

Figure 2-56 Viewing alarm rules



2. Configure long transaction alarms.
 - a. Click  in the upper left corner of the page and choose **Databases > Relational Database Service**.

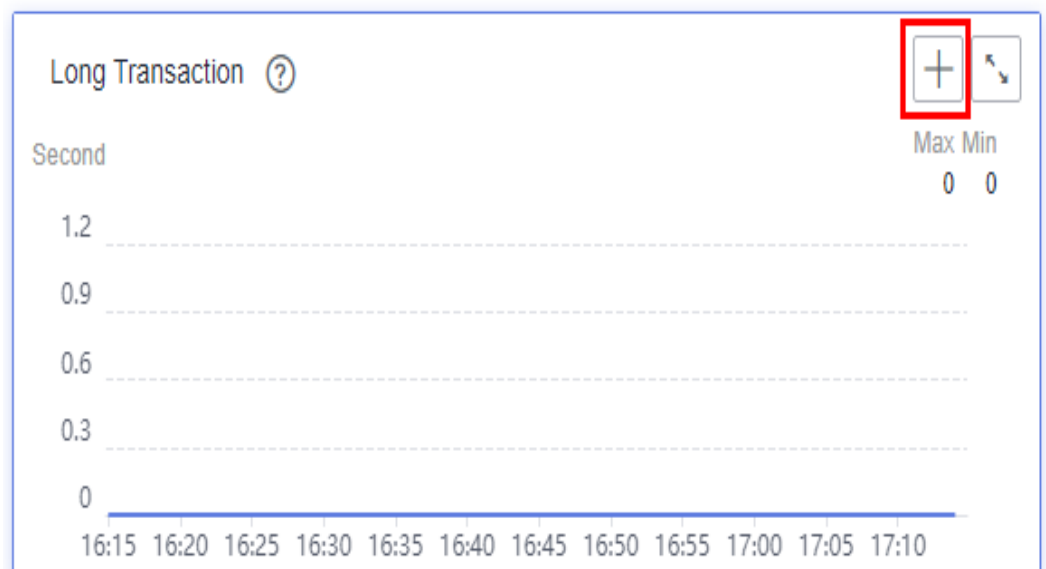
- b. On the **Instances** page, locate the target DB instance and click **View Metrics** in the **Operation** column.
- c. View the **Long Transaction** metric.

Figure 2-57 Viewing metrics

NameID	Description	DB Instance T.	DB Engine Version	Status	Billing Mode	Floating IP Add.	Created	Storage Type	Operation
rds-3923 zdb00b6bc9654a0b9065311695652c4abin01		Single 2 vCPUs 8 GB	MySQL 5.7.41	Available	Pay-per-Use Created on Jul...		Jul 03, 2023 11:11...	Cloud SSD	View Metric Log In More

- d. Click + in the upper right corner of the **Long Transaction** metric.

Figure 2-58 Long Transaction



- e. On the displayed page, set parameters as required. For details about the parameters, see [Creating an Alarm Rule](#).

2.9 Security Best Practices

Security is a shared responsibility between Huawei Cloud and you. Huawei Cloud is responsible for the security of cloud services to provide a secure cloud. As a tenant, you should properly use the security capabilities provided by cloud services to protect data, and securely use the cloud. For details, see [Shared Responsibilities](#).

This section provides actionable guidance for enhancing the overall security of using RDS for MySQL. You can continuously evaluate the security status of your RDS for MySQL DB instances and enhance their overall security defense by

combining different security capabilities provided by RDS for MySQL. By doing this, data stored in RDS for MySQL DB instances can be protected from leakage and tampering both at rest and in transit.

You can make security configurations from the following dimensions to match your workloads.

- [Optimizing Database Connection Configurations to Reduce Network Attack Risks](#)
- [Properly Managing Database Accounts and Passwords to Reduce Data Leakage Risks](#)
- [Strengthening Permissions Management to Reduce Related Risks](#)
- [Enabling Database Audit for Post-Event Backtracking](#)
- [Configuring Data Backup to Ensure Data Reliability](#)
- [Encrypting Data Before Being Stored](#)
- [Hardening Parameter Configuration to Prevent Data Leakage](#)
- [Using the Latest Database Version for Better Experience and Security](#)
- [Using Other Cloud Services for Additional Data Security](#)

Optimizing Database Connection Configurations to Reduce Network Attack Risks

1. **Do not bind an EIP to your RDS for MySQL instance to prohibit unauthorized access and DDoS attacks from the Internet.**

Do not deploy your instance on the Internet or in a demilitarized zone (DMZ). Instead, deploy it on an intranet and use routers or firewalls to control access to your instance. Do not bind an EIP to your instance to prohibit unauthorized access and DDoS attacks from the Internet. If an EIP has been bound to your instance, **unbind it**. If you do need an EIP, configure security group rules to restrict the source IP addresses that can access your instance.

2. **Do not use the default port number.**

RDS for MySQL instances use the default port 3306, leaving your instance more vulnerable to malicious attacks. To avoid this risk, **change the port number** for your DB instance.

3. **Restrict operations of a database user.**

If there is no limit for the resources that a database user can use, the system may be overloaded when the user is attacked, causing a denial of service (DoS) on the system. Setting limitations can prevent excessive resource consumption due to over-utilization of resources. To prevent service availability from being affected in heavy-load scenarios, use the following SQL statements to restrict the number of operations that an individual database user can perform based on your service model:

```
alter user '<user>'@'<hostname>' with max_queries_per_hour <queries_num>;
alter user '<user>'@'<hostname>' with max_user_connections <connections_num>;
alter user '<user>'@'<hostname>' with max_updates_per_hour <updates_num>;
alter user '<user>'@'<hostname>' with max_connections_per_hour
<connections_per_hour>;
```

- *<user>* indicates the username of the account you want to set the limits for.

- *<hostname>* indicates the host name of the account.
 - *<queries_num>* indicates the maximum number of queries allowed for the account per hour.
 - *<connections_num>* indicates the maximum number of concurrent connections allowed for the account.
 - *<updates_num>* indicates the maximum number of updates that the account can issue per hour.
 - *<connections_per_hour>* indicates the maximum number of times the account can connect to the database server per hour.
4. **Do not use the wildcard % for the host name.**
A host name specifies which host is allowed to connect to your database. You can use the **host** field in the **user** table to specify the host. If you enter a wildcard % as the host name, your database is accessible to any IP address, increasing the risk of attacks. To minimize the attack risk, [set the host IP address](#) to a specific network segment or IP address.
 5. **Limit the waiting time of idle database connections.**
Each connection to the MySQL server consumes memory, and the maximum number of connections supported is limited. If the MySQL server has a large number of idle connections, memory consumed by these connections is wasted and the maximum number of connections can be reached. Once the limit is reached, an error message "too many connections" is reported if a new connection is established. You need to set the waiting time for idle connections to ensure that idle connections are cleared in time. Change the values of **wait_timeout** and **interactive_timeout** by referring to [Modifying Parameters of an RDS for MySQL Instance](#).
 6. **Ensure that SSL is enabled by default.**
If SSL is not configured, data transmitted between a MySQL client and server is in plaintext, which is vulnerable to eavesdropping, tampering, and man-in-the-middle attacks. To improve data transmission security, specify the **REQUIRE SSL** attribute for a database account and [configure SSL](#).
You can use the following SQL statements to require SSL connections for a specific account:

```
create user '<user>'@'<hostname>' REQUIRE SSL;  
alter user '<user>'@'<hostname>' REQUIRE SSL;
```

Properly Managing Database Accounts and Passwords to Reduce Data Leakage Risks

1. **Periodically change the password of the administrator.**
The default database administrator account **root** has high permissions. You are advised to periodically change the password of user **root** by referring to [Resetting the Administrator Password to Restore Root Access](#).
2. **Configure password complexity.**
As a collector of information, a database system is easy to be the target of attacks. You need to keep your database account and password secure. In addition, configure the complexity of your password to avoid using weak passwords. For details, see "Setting Password Complexity" in [Database Account Security](#).

3. Configure a password expiration policy.

Using the same password too long makes it easier for hackers to crack or guess your password. To prevent this, [configure a password expiration policy](#) to limit how long a password can be used.

Strengthening Permissions Management to Reduce Related Risks

1. Do not create stored procedures or functions as the administrator.

Stored procedures and functions are run as creators by default. If you create stored procedures and functions as the administrator, regular users can run them through privilege escalation, so do not use the administrator account to create stored procedures or functions.

2. Review and harden permission configurations.

Check whether the following permission configurations meet security requirements. If they do not meet security requirements, harden the security configuration.

- Ensure that only the administrator account can perform operations on the **mysql.user** table.
- Ensure that the **Process_priv** permission can be granted only to the administrator account.
- Ensure that the **Create_user_priv** permission can be granted only to the administrator account.
- Ensure that the **Grant_priv** permission can be granted only to the administrator account.
- Ensure that the **Reload_priv** permission can be granted only to the administrator account.
- Ensure that the replication account has only the **replication slave** permission.
- Ensure that the database metric monitoring account has only the **replication client** permission.

Example: If a non-administrator account has the **Process** permission, run the following SQL statement to revoke this permission:

```
revoke process on *.* from <your_account>;
```

In the preceding statement, *<your_account>* indicates the username of the account whose **Process** permission needs to be revoked.

Enabling Database Audit for Post-Event Backtracking

The database audit function records all user operations on the database in real time. This function logs, analyzes, and reports user activities in the database. Based on the audit logs, you can prepare compliance reports and track incidents, improving data asset security. For details, see [Enabling SQL Audit](#).

Configuring Data Backup to Ensure Data Reliability

1. Enable data backup.

RDS for MySQL supports automated and manual backups. You can periodically back up databases. If a database is faulty or data is damaged, you

can restore the database using backups to ensure data reliability. For details, see [Data Backups](#).

2. **Configure a binlog clearing policy.**

Binlogs continuously increase as services run. You need to configure a clearing policy to prevent disk expansion. [Set a retention period for RDS for MySQL binlogs](#).

Encrypting Data Before Being Stored

To improve data security, [enable server-side encryption](#). After it is enabled, data will be encrypted on the server before being stored when you create a DB instance or scale up storage space. This reduces the risk of data leakage.

Hardening Parameter Configuration to Prevent Data Leakage

1. **Set `local_infile` to OFF.**

If `local_infile` is set to **ON**, a database client can use the **load data local** syntax to load local files to database tables. For example, when a web server functions as a database client to connect to a database, if the web server has an SQL injection vulnerability, an attacker can use the **load data local** command to load sensitive files on the web server to the database, causing information leakage. To prevent this, set `local_infile` to **OFF** by referring to [Modifying Parameters of an RDS for MySQL Instance](#).

2. **Set `sql_mode` to `STRICT_ALL_TABLES`.**

When attempting to launch an attack, an attacker may enter various parameters in a trial-and-error manner. If the server adapts to incorrect statements, database data may be leaked. Therefore, **STRICT_ALL_TABLES** is recommended. Even if an error occurs in other rows than the first row, the statement will be discarded once an invalid data value is found. This method maximally ensures that database information is not disclosed. You are advised to set `sql_mode` to **STRICT_ALL_TABLES** by referring to [Modifying Parameters of an RDS for MySQL Instance](#).

Using the Latest Database Version for Better Experience and Security

The MySQL community irregularly discloses newly discovered vulnerabilities. RDS for MySQL evaluates the actual risks of database kernel versions and release new database kernel versions accordingly. To improve the usability and security of the database system, you are advised to use [the latest database version](#).

Using Other Cloud Services for Additional Data Security

To obtain extended data security capabilities, you are advised to use [Database Security Service \(DBSS\)](#).

3 RDS for PostgreSQL

3.1 Creating a Cross-Region DR Relationship for an RDS for PostgreSQL Instance

3.1.1 Overview

Scenarios

You can create a cross-region DR relationship for your DB instance. If the production instance fails or the service system breaks down due to other force majeure factors, the DR instance in another region ensures that the production data is not lost and the production system continues to run without interruption. This enhances system availability.

This practice includes the following tasks:

- Create an RDS for PostgreSQL instance.
- Create a cross-region DR relationship for the RDS for PostgreSQL instance.

Prerequisites

- You have registered with Huawei Cloud and completed real-name authentication.
- Your account balance is greater than or equal to \$0 USD.

Constraints

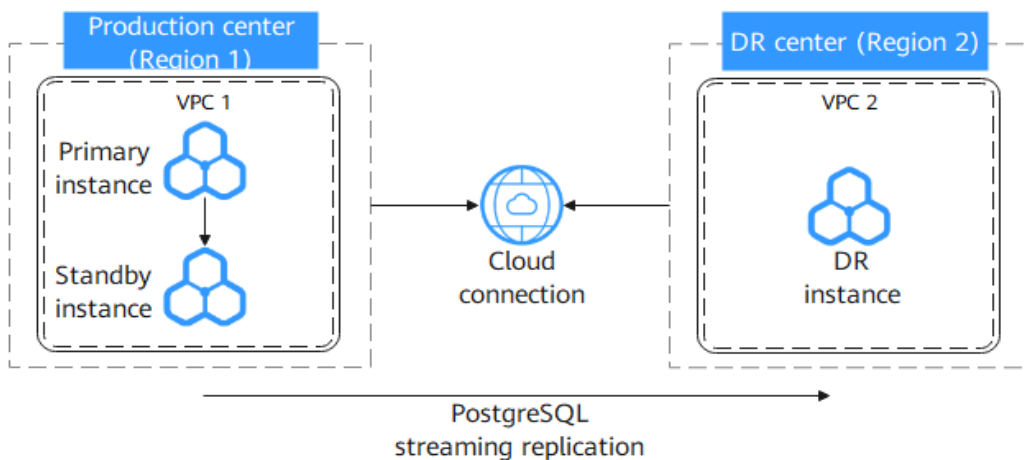
- The primary DB instance and DR instance are running properly and are deployed in different clouds or regions. The primary DB instance is deployed in primary/standby mode and the DR instance is deployed in standalone mode.
- Before configuring disaster recovery for the DR instance, you must configure it for the primary instance. Otherwise, the DR relationship cannot be established.

- The specifications of the DR instance are at least equal to those of the primary DB instance.
- Cross-cloud or cross-region DR is supported only for RDS for PostgreSQL 12 and later versions.
- Cross-cloud or cross-region DR relationships cannot be established across major versions.
- The DR instance can be promoted to primary and the DR replication status can be queried only after the DR relationship between the primary DB instance and DR instance is established.
- Ensure that the primary DB instance and DR instance are in the regions where Cloud Connect or Virtual Private Network (VPN) has been rolled out.
- DR instances do not support point-in-time recovery (PITR) or CBR snapshot-based backups. Perform such operations on the primary instance if needed.

How Cross-Region DR Works

Two RDS for PostgreSQL instances are deployed in two data centers, one in the production center and the other in the DR center. RDS replicates data from the primary instance in the production center to the DR instance in the DR center, keeping data synchronous across the regions. Before using this function, ensure that Cloud Connect can be used to connect the two regions.

Figure 3-1 Diagram



Service List

- Cloud Connect
- Virtual Private Cloud (VPC)
- Relational Database Service (RDS)

Notes on Usage

- The resource planning in this practice is for demonstration only. Adjust it as needed.
- The end-to-end test data in this practice is for reference only.

3.1.2 Resource Planning

Table 3-1 Resource planning

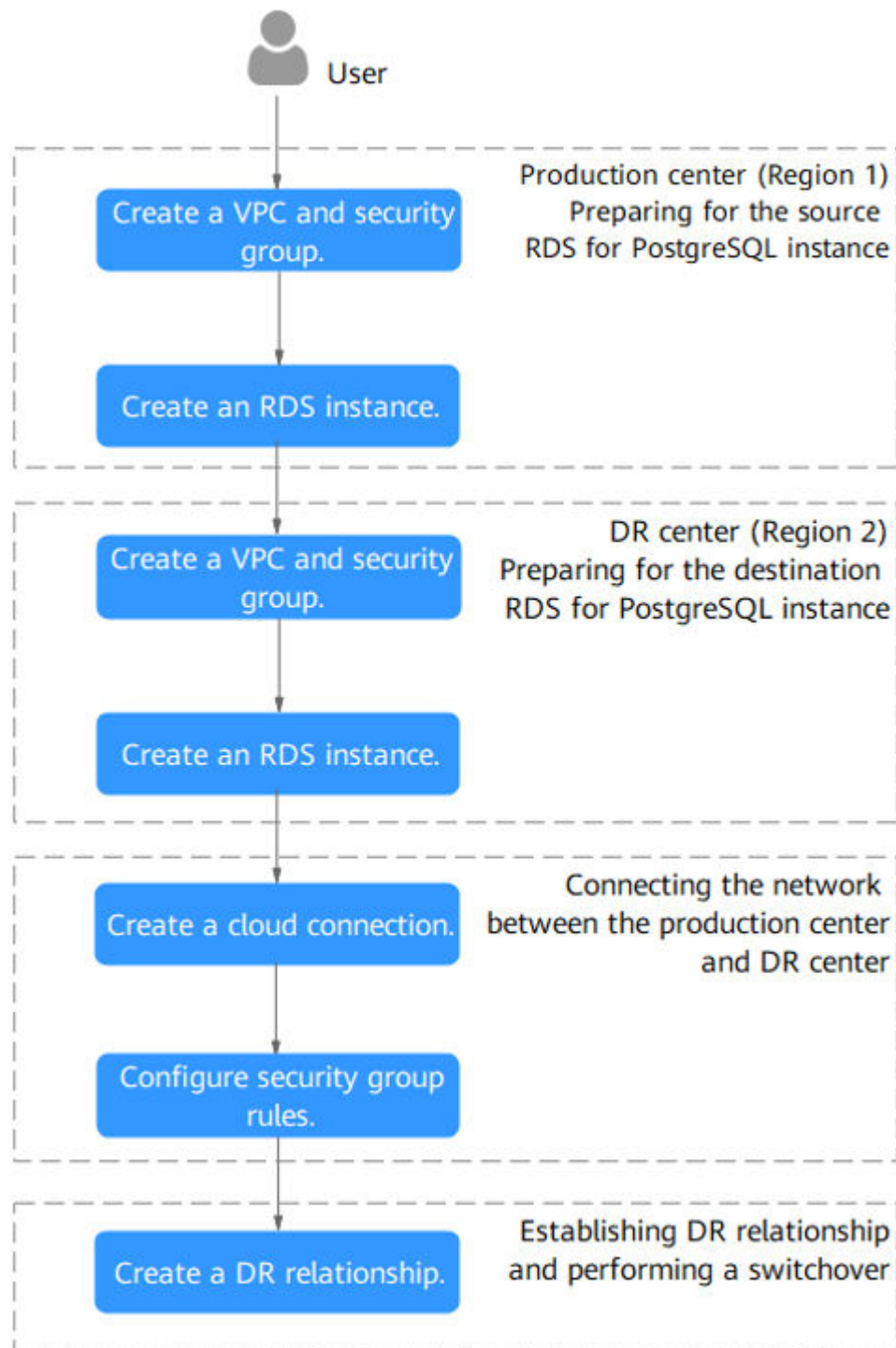
Category	Subcategory	Planned Value	Description
Production center VPC	VPC name	vpc-pg-01	Specify a name that is easy to identify.
	Region	CN-Hong Kong	To reduce network latency, select the region nearest to you.
	AZ	az1	-
	Subnet	192.168.10.0/24	Select a subnet with sufficient network resources.
	Subnet name	subnet-2aa1	Specify a name that is easy to identify.
DR center VPC	VPC name	vpc-pg-02	Specify a name that is easy to identify.
	Region	AP-Singapore	To reduce network latency, select the region nearest to you.
	AZ	az1	-
	Subnet	192.168.20.0/24	Select a subnet with sufficient network resources.
	Subnet name	subnet-a388	Specify a name that is easy to identify.
RDS for PostgreSQL instance in the production center	DB instance name/ID	rds-pg-01 04**in03	Specify a name that is easy to identify.
	Region	CN-Hong Kong	To reduce network latency, select the region nearest to you.
	DB engine version	PostgreSQL 12	-
	Private IP address	192.168.10.117	-
	DB instance type	Primary/Standby	Select Primary/Standby for the production instance.
	Storage type	Cloud SSD	-
	AZ	az1, az3	-

Category	Subcategory	Planned Value	Description
	Instance class	Dedicated 2 vCPUs 4 GB	-
	Storage space	40 GB	-
RDS for PostgreSQL instance in the DR center	DB instance name/ID	rds-pg-025f**in03	Specify a name that is easy to identify.
	Region	AP-Singapore	To reduce network latency, select the region nearest to you.
	DB engine version	PostgreSQL 12	-
	Private IP address	192.168.20.69	-
	DB instance type	Single	Select Single for the DR instance.
	Storage type	Cloud SSD	-
	AZ	az1	-
	Instance class	Dedicated, 8 vCPUs 16 GB	The CPU and memory specifications of the DR instance must be greater than or equal to those of the primary instance.
	Storage space	100 GB	The storage space of the DR instance must be greater than or equal to that of the primary instance.

3.1.3 Operation Process

The following figure shows the process of creating an RDS for PostgreSQL production instance and a DR instance and how to migrate data from the production instance to the DR instance.

Figure 3-2 Flowchart



3.1.4 Preparing an RDS for PostgreSQL Instance in the Production Center

This section describes how to create a VPC, a security group, and an RDS for PostgreSQL instance in the production center.

- **Step 1: Create a VPC and Security Group**

- **Step 2: Create an RDS for PostgreSQL Instance**

Step 1: Create a VPC and Security Group

Step 1 Go to the [Create VPC](#) page.

Step 2 Select **CN-Hong Kong** for **Region**. Configure the basic information, subnet, and IP address.

Figure 3-3 Creating a VPC

Step 3 Click **Create Now**.

Step 4 In the navigation pane of **Network Console**, choose **Access Control > Security Groups**.

Step 5 Click **Create Security Group**.

Figure 3-4 Creating a security group

Step 6 Click **Create Now**.

----End

Step 2: Create an RDS for PostgreSQL Instance

Step 1 Go to the [Buy DB Instance](#) page.

Step 2 Select **CN-Hong Kong** for **Region**. Configure the instance information and click **Buy**.

Figure 3-5 Selecting a DB engine version

The screenshot shows the 'Buy DB Instance' configuration page. Key settings include:

- Billing Mode:** Pay-per-use (selected)
- Region:** CN East-Shanghai1
- Project:** CN East-Shanghai1
- DB Instance Name:** rds-bdf6
- DB Engine:** PostgreSQL (selected)
- DB Engine Version:** 12 (selected)
- DB Instance Type:** Primary/Standby (selected)
- Storage Type:** Cloud SSD (selected)
- Primary AZ:** AZ1
- Standby AZ:** AZ3
- Time Zone:** (UTC+08:00) Beijing, Chongqing, Hon...

Figure 3-6 Selecting an instance class

The screenshot shows the 'Instance Class' selection page. Key settings include:

- Instance Class:** Dedicated (selected)
- vCPUs | Memory:** 4 vCPUs | 32 GB (selected)
- Recommended Connections:** 3,200
- Storage Space:** 40 GB
- Autoscaling:** Disabled
- Disk Encryption:** Disabled

Figure 3-7 Configuring network information as planned

The screenshot shows the configuration page for an RDS instance. The VPC is set to 'vpc-pg-01' and the subnet is 'subnet-53dd192:168.0.0/24'. A note states: 'The VPC an RDS instance is deployed in cannot be changed later. ECSs in different VPCs cannot communicate with each other by default. If you want to create a VPC, go to the VPC console. An EIP is required if you want to access DB instances through a public network. View EIP'. The Security Group is set to 'default'. There are links for 'View In-use IP Addresses (Addresses available: 251)', 'View Security Group', 'Create Security Group', and 'Security Group Rules'.

Figure 3-8 Setting an administrator password

The screenshot shows the 'Setting an administrator password' step. It includes a 'Password' section with 'Configure' and 'Skip' buttons. The 'Administrator' is set to 'root'. The 'Administrator Password' and 'Confirm Password' fields are masked with dots. A note says: 'Keep your password secure. The system cannot retrieve your password.' Below this are 'Parameter Template' (Default-PostgreSQL-12) and 'Enterprise Project' (default) dropdowns. The 'Tag' section has a text area for adding tags and buttons for 'Enter a tag key', 'Enter a tag value', and 'Add'. A note says: 'Predefined tags are recommended for adding the same tag to different cloud resources. Create Predefined Tag View Predefined Tags. To add a tag, enter a tag key and a tag value below. You can add 20 tags more tags.' At the bottom, the 'Quantity' is set to 1, with a note: 'You can create 49 more instances (read replicas included). Increase Quota'.

Step 3 Confirm the settings.

- To modify your settings, click **Previous**.
- If you do not need to modify your settings, click **Submit**.

----End

3.1.5 Preparing an RDS for PostgreSQL Instance in the DR Center

This section describes how to create a VPC, a security group, and an RDS for PostgreSQL instance in the DR center.

NOTICE

- The VPC subnet CIDR block of the DR instance must be different from that of the production instance. This is the prerequisite for cross-region network connection.
 - The security groups in the production center and DR center must allow access from the database ports in the VPC subnet CIDR blocks to each other.
-
- [Step 1: Create a VPC and Security Group](#)
 - [Step 2: Create an RDS for PostgreSQL Instance](#)

Step 1: Create a VPC and Security Group

Step 1 Go to the [Create VPC](#) page.

Step 2 Select **AP-Singapore** for **Region**. Configure the basic information, subnet, and IP address.

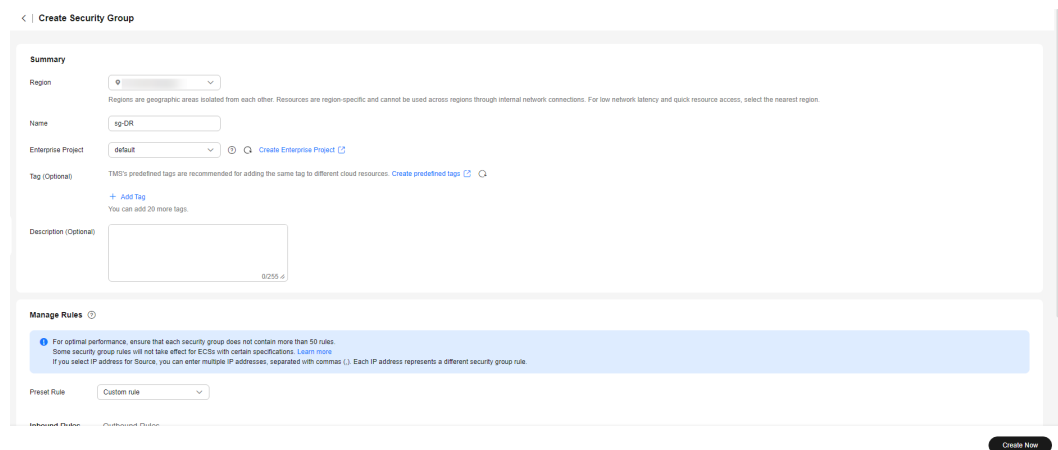
Figure 3-9 Creating a VPC

Step 3 Click **Create Now**.

Step 4 In the navigation pane of **Network Console**, choose **Access Control > Security Groups**.

Step 5 Click **Create Security Group**.

Figure 3-10 Creating a security group



Step 6 Click **Create Now**.

-----End

Step 2: Create an RDS for PostgreSQL Instance

Step 1 Go to the **Buy DB Instance** page.

Step 2 Select **AP-Singapore** for **Region**. Configure the instance information and click **Buy**.

Figure 3-11 Selecting a DB engine version

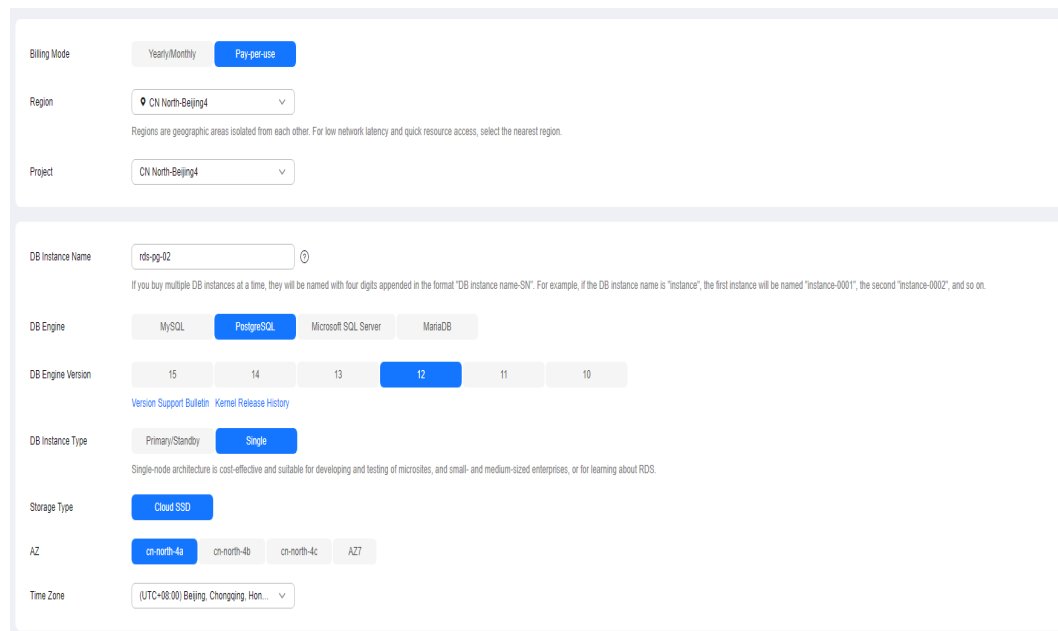


Figure 3-12 Selecting an instance class

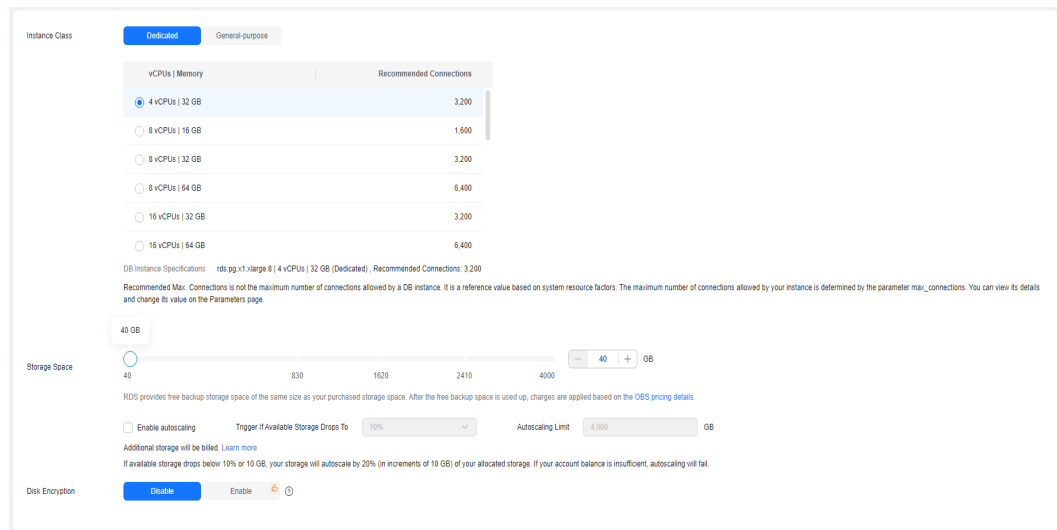


Figure 3-13 Configuring network information as planned

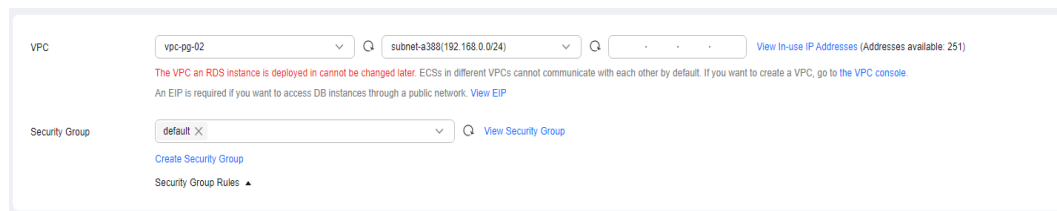
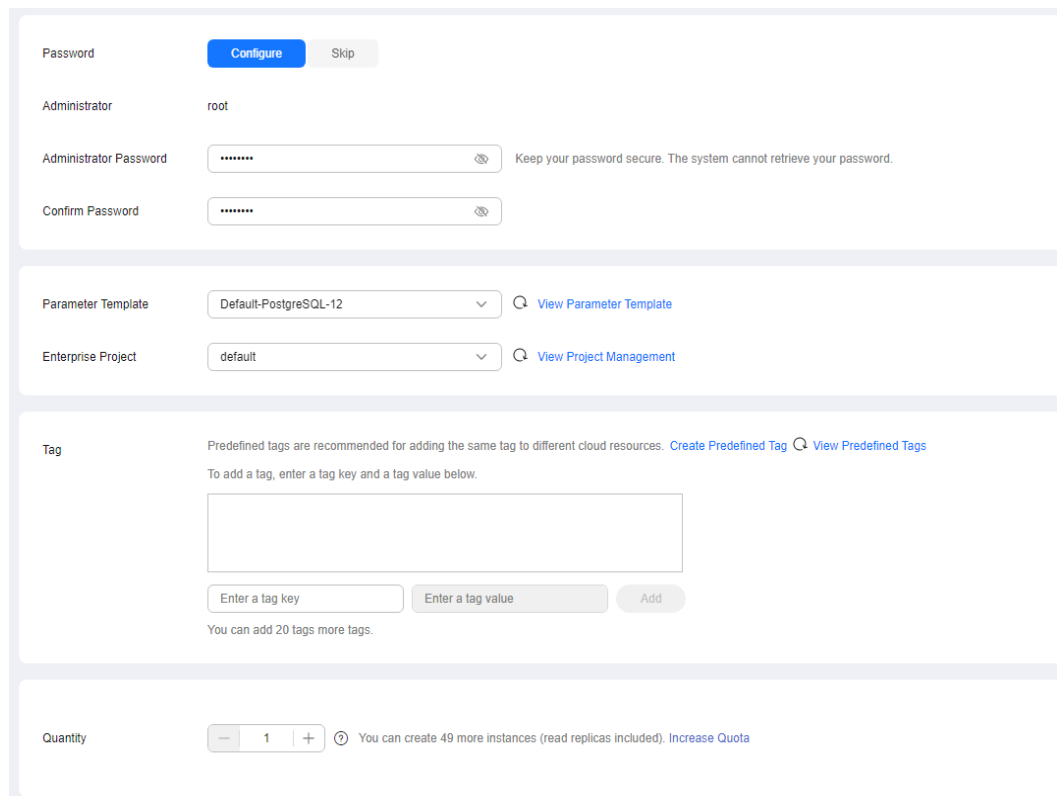


Figure 3-14 Setting an administrator password



Step 3 Confirm the settings.

- To modify your settings, click **Previous**.
- If you do not need to modify your settings, click **Submit**.

----End

3.1.6 Configuring Cross-Region Network Connectivity

Before setting up a DR relationship, you need to configure cross-region network connectivity. For details, see [Method 1: Using Cloud Connect to Connect VPCs in Different Regions](#) or [Method 2: Using VPN to Connect VPCs in Different Regions](#).

You are advised to set a bandwidth size based on the transaction log generation rate metric. The bandwidth must be greater than or equal to 10 times the maximum value of this metric. This is because the unit of the network bandwidth is Mbit/s and that of the transaction log generation rate is MB/s.

For example, if the maximum transaction log generation rate is 10 MB/s, you are advised to set network bandwidth to 100 Mbit/s so that it is sufficient enough for the DR instance to synchronize data from the primary instance in a timely manner.

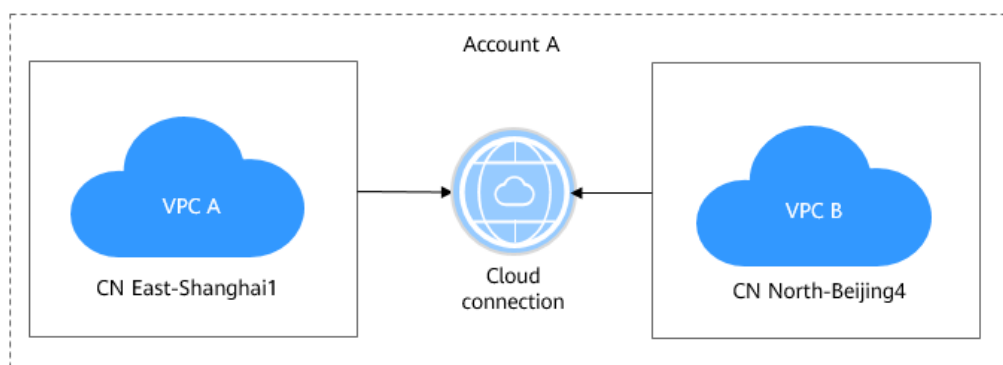
After the network is connected, you need to configure the security groups for the primary instance and DR instance to allow traffic from each other. For details, see [Configuring Security Groups](#).

Method 1: Using Cloud Connect to Connect VPCs in Different Regions

Before setting up a DR relationship, you need to configure cross-region network connectivity.

You can use [Cloud Connect](#) to connect VPCs across regions.

Figure 3-15 Communication between VPCs in the same account but different regions



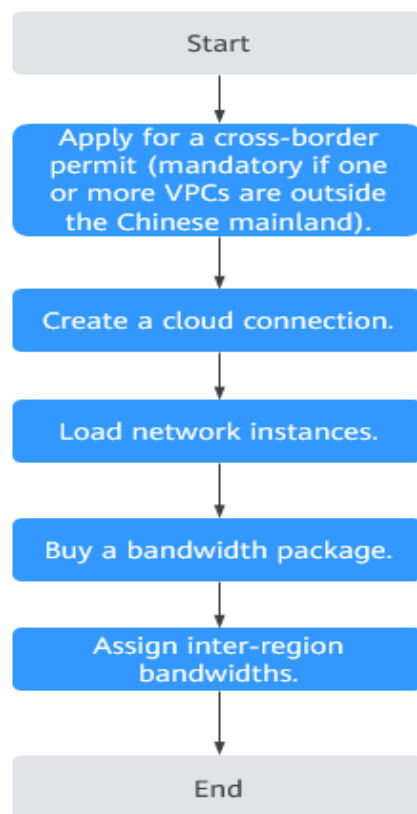
NOTICE

Ensure that the primary and DR instances are in the [regions where cloud connections are available](#).

Ensure that the VPC subnets to which the primary and DR instances belong allow access from each other.

For details about how to enable communication between VPCs in different regions, see [Using a Cloud Connection to Connect VPCs in Different Regions](#).

Figure 3-16 Flowchart



Method 2: Using VPN to Connect VPCs in Different Regions

You can use [Virtual Private Network \(VPN\)](#) to enable communication between VPCs across regions.

NOTICE

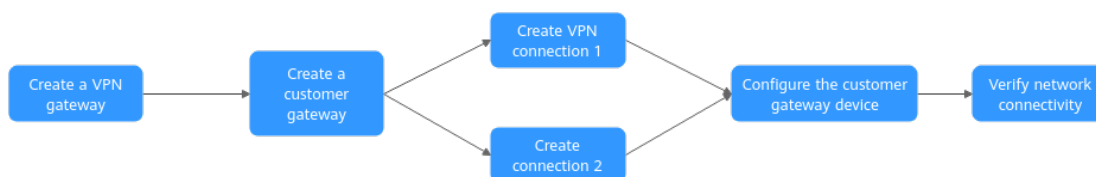
Ensure that the primary and DR instances are in the [regions where VPN is available](#).

After configuring the VPN service, you need to contact the VPN customer service to configure the network.

Ensure that the VPC subnets to which the primary and DR instances belong allow access from each other.

For details about how to configure a VPN connection, see [Overview](#).

Figure 3-17 Flowchart



Configuring Security Groups

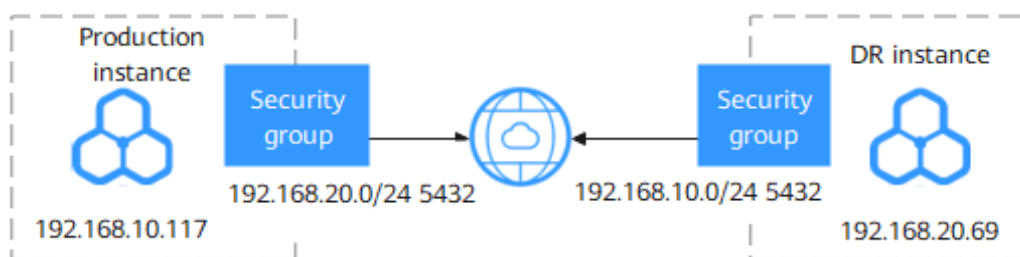
After connecting two VPCs in different regions, you need to configure security groups for the primary and DR instances so that ports in different VPC CIDR blocks can communicate with each other.

Suppose that there are two instances listed in [Table 3-2](#) and they use the default port 5432. The firewall configurations for them are as shown in [Figure 3-18](#).

Table 3-2 CIDR block configurations

Instance	VPC CIDR Block	IP Address
Production instance	192.168.10.0/24	192.168.10.117
DR instance	192.168.20.0/24	192.168.20.69

Figure 3-18 Firewall configurations



3.1.7 Creating a DR Relationship

Scenarios

After a cross-region DR relationship is created, if the region where the primary instance is located encounters a natural disaster and the primary instance cannot be connected, you can promote the DR instance in another region to primary. To connect to the new primary instance, you only need to change the connection address on the application side.

Precautions

- Before using this function, ensure that the network between the DB instances in two different regions is connected. You can use [Cloud Connect](#) or [Virtual Private Network \(VPN\)](#) to connect the VPCs in different regions.
- Before using this function, ensure that the primary instance and DR instance are available and are deployed in different regions. The primary instance uses a primary/standby deployment and the DR instance uses a standalone deployment.
- The vCPUs, memory, and storage space of the DR instance must be greater than or equal to those of the primary instance.
- RDS for PostgreSQL 12 and later versions support cross-region DR.
- After changing the database port or private IP address of the primary instance, you need to re-establish the DR relationship.

Procedure



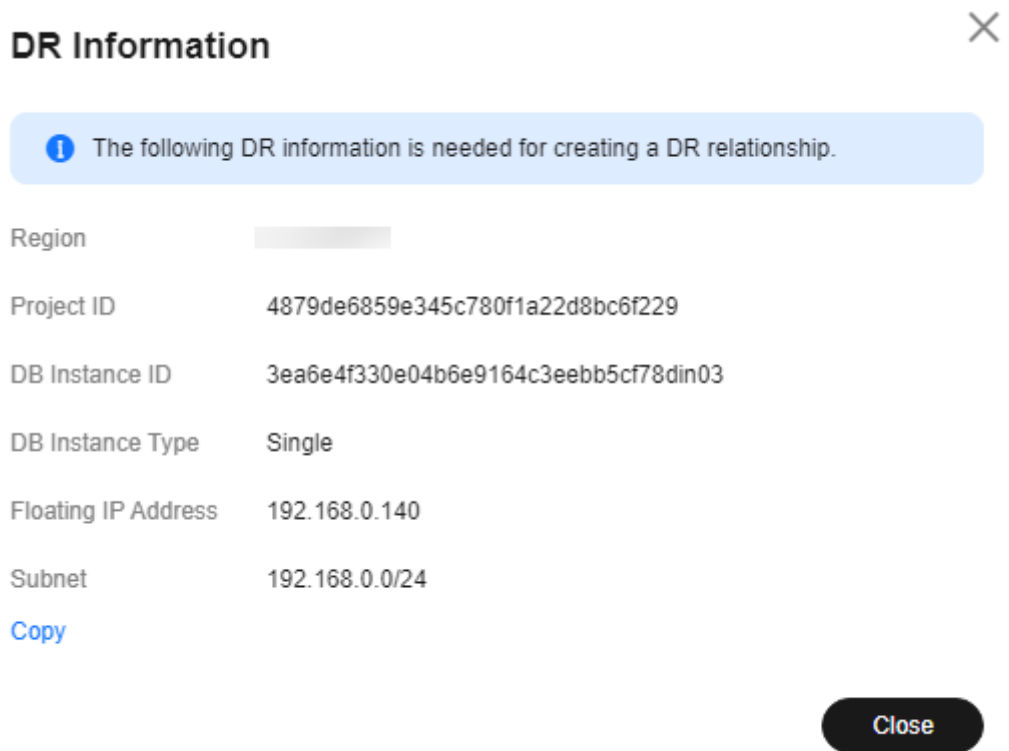
- Step 1** Paste the configuration information of the DR instance to the production instance and configure DR for the production instance.
1. [Log in to the management console](#).
 2. Click  in the upper left corner and select the region where the DR instance is located, for example, AP-Singapore.
 3. Click  in the upper left corner of the page and choose **Databases > Relational Database Service**.
 4. On the **Instances** page, click the DR instance name to go to the **Overview** page.
 5. Click **DR Information**.
 6. In the displayed dialog box, click **Copy**.

Figure 3-19 Copying DR instance configuration information




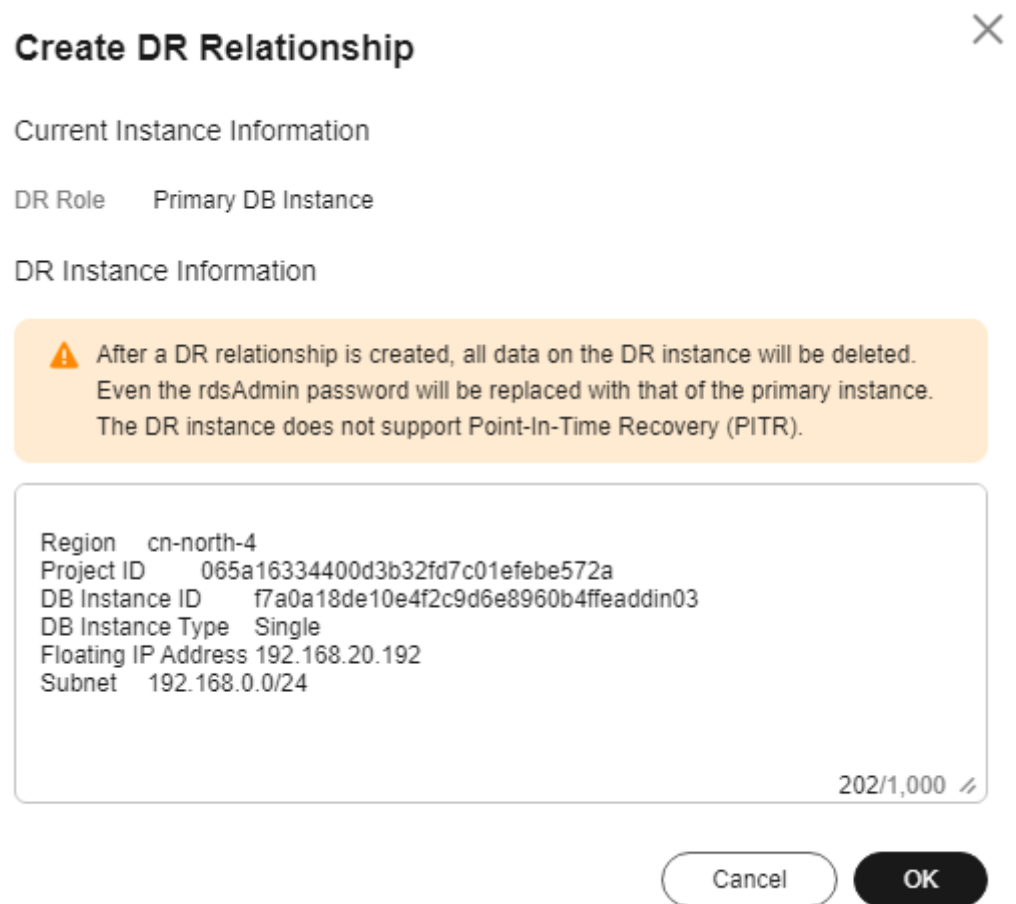
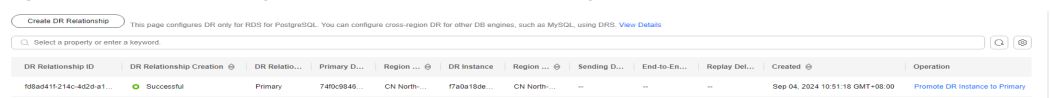
7. Click  in the upper left corner of the management console and select the region where the production instance is located, for example, CN-Hong Kong.
8. On the **Instances** page, locate the production instance and choose **More > View DR Details** in the **Operation** column.
9. Click **Create DR Relationship**. In the displayed dialog box, paste the DR information copied from [Step 1.6](#) to the text box and click **OK** to configure DR for the production instance.


Figure 3-20 Pasting DR instance information



10. On the **DR Management** page of the production instance, check whether the DR is configured. If the value of **DR Relationship Creation** is **Successful**, the DR is successfully configured for the production instance. Perform subsequent operations only after this task is successfully executed.

Figure 3-21 Checking whether the DR is configured



- Step 2** Paste the information of the production instance to the DR instance and configure DR for the DR instance.
1. On the **Instances** page, click the production instance name to go to the **Overview** page.
 2. Click **DR Information**.
 3. In the displayed dialog box, click **Copy**.
 4. Click  in the upper left corner and select the region where the DR instance is located, for example, AP-Singapore.
 5. On the **Instances** page, locate the DR instance and choose **More > View DR Details** in the **Operation** column.

6. Click **Create DR Relationship**. In the displayed dialog box, paste the DR information copied from [Step 2.3](#) to the text box and click **OK** to configure DR for the DR instance.
7. On the **DR Management** page, check whether the DR is configured. If the value of **DR Relationship Creation** is **Successful**, the DR is successfully configured for the DR instance.
8. On the **DR Management** page, you can view the DR replication status, sending delay, end-to-end delay, and replay delay.

----End

3.1.8 Promoting a DR Instance to Primary



Scenarios

If the region where the primary instance is located suffers a natural disaster and the primary instance cannot be connected, you can promote the DR instance in another region to primary. To connect to the new primary instance, you only need to change the connection address on the application side.

Precautions

After a DR instance is promoted to primary, the DR relationship between it and the original primary instance is removed.

Procedure

- Step 1** [Log in to the management console](#).
- Step 2** Click  in the upper left corner and select the region where the DR instance is located, for example, AP-Singapore.
- Step 3** Click  in the upper left corner of the page and choose **Databases > Relational Database Service**.
- Step 4** On the **Instances** page, click the name of the DR instance.
- Step 5** In the navigation pane, choose **DR Management**.
- Step 6** In the DR relationship list, click **Promote DR Instance to Primary** in the **Operation** column.
- Step 7** In the displayed dialog box, click **OK**.
- Step 8** Check the task execution result on the **Task Center** page. If the task status is **Completed**, the promotion is successful.
- Step 9** Change the database connection address on the application side and manually switch workloads over to the new primary instance.

----End

3.1.9 Removing a DR Relationship

Scenarios

If a DR relationship is no longer needed, you can remove it.

Precautions

Only the DR relationship that has been successfully established can be removed. You must first remove the DR relationship of the DR instance and then that of the primary instance. Otherwise, an alarm may be generated.

Procedure

Step 1 [Log in to the management console.](#)

Step 2 Remove the DR relationship of the DR instance.

1. On the **Overview** page of the primary instance, click **DR Information**.
2. In the displayed dialog box, click **Copy**.
3. On the **Instances** page, click the name of the DR instance.
4. In the navigation pane, choose **DR Management**.
5. In the DR relationship list, click **Remove DR Relationship** in the **Operation** column.
6. Paste the copied DR information to the dialog box.
7. Check the task execution result on the **DR Management** page. If the list is deleted, the task is successfully executed.

Step 3 Remove the DR relationship of the primary instance by referring to [Step 2](#).

----End

3.1.10 FAQs

- Question 1: What should I do if I fail to configure DR for a DR instance?
Check whether the security groups of the production center and DR center allow traffic from the ports of the VPC subnets each other. If a VPN connection is used, check whether the VPCs are connected as instructed in [Method 2: Using VPN to Connect VPCs in Different Regions](#).
- Question 2: Why did the system display a message indicating that there is a route conflict in the current VPC when I use Cloud Connect to connect VPCs?
Rectify the fault by referring to [What Can I Do If There Is a Route Conflict When I Load a Network Instance to a Cloud Connection?](#)

3.2 RDS for PostgreSQL Publications and Subscriptions

Logical Definition

A publication can be defined on any primary physical replication server. The node where a publication is defined is called the publisher. A publication is a set of

changes generated from a table or a group of tables, and might also be described as a change set or replication set. Each publication exists in only one database.

A subscription is the downstream side of logical replication. The node where a subscription is defined is called the subscriber. A subscription defines the connection to another database and the set of publications (one or more) to which it wants to subscribe. The subscriber database behaves in the same way as any other RDS for PostgreSQL instance (primary) and can be used as a publisher for other databases by defining its own publications.

Required Permissions

- To create a publication, the publisher must have the replication permission.
- When creating a publication for ALL TABLES, ensure that the publisher uses the **root** user of the initial or later versions for privilege escalation.
- When creating or deleting a subscription, ensure that the subscriber uses the **root** user of the initial or later versions for privilege escalation.
- When creating a publication or subscription, ensure that the publisher and subscriber are in the same VPC.

For details about **root** privilege escalation of each version, see [Privileges of the root User](#).

Restrictions on Publications

- Publications may currently only contain tables (indexes, sequence numbers, and materialized views cannot be published). Each table can be added to multiple publications if needed.
- One publication can have multiple subscribers.
- ALL TABLES can be used to publish all tables.
- Multiple publications can be created in a given database, but each publication must have a unique name. The created publications can be obtained by querying pg_publication.
- Publications can choose to limit the changes they produce to any combination of INSERT, UPDATE, DELETE, and TRUNCATE, similar to how triggers are fired by particular event types. By default, all operation types are replicated.

Example: To publish the UPDATE and DELETE operations on the **t1** table:

```
CREATE PUBLICATION update_delete_only FOR TABLE t1
WITH (publish = 'update, delete');
```

- Replica identity: A published table must have a replica identity configured in order to be able to replicate UPDATE and DELETE operations. If **nothing** is set for the replica identity, subsequent UPDATE or DELETE operations will cause an error on the publisher.

You can obtain the replica identity of a table from **pg_class.relreplident**.

relreplident is of character type and identifies columns used to form "replica identity" for rows: d = default, f = all columns, i = index, and n = nothing.

To check whether a table has an index constraint that can be used as a replica identity, run the following:

```
SELECT quote_ident(nspname) || '.' || quote_ident(relname) AS name, con.ri AS keys,
CASE relreplident WHEN 'd' THEN 'default' WHEN 'n' THEN 'nothing' WHEN 'f' THEN
'full' WHEN 'i' THEN 'index' END AS replica_identity
```

```
FROM pg_class c JOIN pg_namespace n ON c.relnamespace = n.oid, LATERAL (SELECT
array_agg(contype) AS ri FROM pg_constraint WHERE conrelid = c.oid) con
WHERE relkind = 'r' AND nspname NOT IN ('pg_catalog', 'information_schema', 'monitor',
'repack', 'pg_toast')
ORDER BY 2,3;
```

- Command for changing a replica identity

The replica identity of a table can be changed using **ALTER TABLE**.

```
ALTER TABLE table_name REPLICA IDENTITY
{ DEFAULT | USING INDEX index_name | FULL | NOTHING };
```

-- There are four forms:

```
ALTER TABLE t_normal REPLICA IDENTITY DEFAULT;           -- The primary key is
used as the replica identity. If there is no primary key, the replica identity is set to FULL.
ALTER TABLE t_normal REPLICA IDENTITY FULL;               -- The entire row is used
as the replica identity.
ALTER TABLE t_normal REPLICA IDENTITY USING INDEX t_normal_v_key; -- A unique index
is used as the replica identity.
ALTER TABLE t_normal REPLICA IDENTITY NOTHING;           -- No replica identity is
set.
```

- Precautions for using replica identities
 - If a table has a primary key, the replica identity can be set to **DEFAULT**.
 - If a table does not have a primary key but has a non-null unique index, the replica identity can be set to **INDEX**.
 - If a table does not have a primary key or a non-null unique index, the replica identity can be set to **FULL**. This, however, is very inefficient and should only be used as a fallback if no other solution is possible.
 - In all cases other than those mentioned above, logical replication cannot be implemented. The output information is insufficient, and an error may be reported.
 - **If a table with replica identity "nothing" is added to logical replication, deleting or updating the table will cause an error on the publisher.**

Restrictions on Subscriptions

- To ensure that failover slots are used, failover slots must be created on the publisher and associated with the existing replication slots using **create_slot = false**.

```
CREATE SUBSCRIPTION sub1 CONNECTION 'host=192.168.0.1 port=5432
user=user1 dbname=db1' PUBLICATION pub_name with (create_slot =
false,slot_name = FailoverSlot_name);
```
- Logical replication does not replicate DDL changes, so the tables in the publication set must already exist on the subscriber.
- Multiple subscriptions can be created in a given database. These subscriptions can come from one or more publishers.
- A given table of a subscriber cannot accept multiple publications from the same source.
- When creating a subscription or altering a subscription, you can use **enable** to enable the subscription or **disable** to suspend the subscription.
- To delete a subscription, use **DROP SUBSCRIPTION**. Note that after a subscription is deleted, the local table and data are not deleted, but upstream information of the subscription is no longer received.

NOTICE

If a subscription is associated with a replication slot, **DROP SUBSCRIPTION** cannot be executed inside a transaction block. You can use **ALTER SUBSCRIPTION** to disassociate the subscription from the replication slot.

To completely delete a subscription, perform the following steps:

- a. Query the replication slot associated with the subscription on the subscriber.

```
select subname,subconninfo,subslotname from pg_subscription where subname = 'sub2';
```

- **subname** indicates the subscriber name.
 - **subconninfo** indicates information about the connected remote host.
 - **subslotname** indicates the replication slot name of the remote host.
- b. On the subscriber, disassociate the subscription from the replication slot and delete the subscription.

```
ALTER SUBSCRIPTION subname SET (slot_name = NONE);  
DROP SUBSCRIPTION subname;
```

- c. Delete the associated replication slot at the publisher.

```
select pg_drop_replication_slot(' slot_name');
```

Syntax Reference

- Publications

CREATE PUBLICATION is used to create a publication, **DROP PUBLICATION** is used to delete a publication, and **ALTER PUBLICATION** is used to modify a publication.

After a publication is created, tables can be added or removed dynamically using **ALTER PUBLICATION**. Such operations are all transactional.

- Subscriptions

CREATE SUBSCRIPTION is used to create a subscription, **DROP SUBSCRIPTION** is used to delete a subscription, and **ALTER SUBSCRIPTION** is used to modify a subscription.

After creating a subscription, you can use **ALTER SUBSCRIPTION** to suspend or resume the subscription at any time. Deleting and recreating a subscription results in the loss of synchronized information, which means that related data needs to be synchronized again.

For details, see the official documentation. PostgreSQL 13 is used as an example.

- Creating a publication: <https://www.postgresql.org/docs/13/sql-createpublication.html>
- Deleting a publication: <https://www.postgresql.org/docs/13/sql-droppublication.html>
- Modifying a publication: <https://www.postgresql.org/docs/13/sql-alterpublication.html>

3.3 User-Defined Data Type Conversion

Description

There are three data type conversion modes for PostgreSQL: implicit conversion, assignment conversion, and explicit conversion. They correspond to i (Implicit), a (Assignment), and e (Explicit) in the `pg_cast` system catalog.

- Implicit conversion: a conversion from low bytes to high bytes of the same data type, for example, from **int** to **bigint**
- Assignment conversion: a conversion from high bytes to low bytes of the same data type, for example, from **smallint** to **int**
- Explicit conversion: a conversion between different data types

How to Use

1. Before converting data types, you can run the following command to check whether RDS for PostgreSQL supports data type conversion:

```
select * from pg_catalog.pg_cast ;
oid | castsource | casttarget | castfunc | castcontext | castmethod
-----+-----+-----+-----+-----+-----
11277 | 20 | 21 | 714 | a | f
11278 | 20 | 23 | 480 | a | f
11279 | 20 | 700 | 652 | i | f
11280 | 20 | 701 | 482 | i | f
.....
```

2. Run the following command to check whether **int4** can be converted to **text**:

```
select * from pg_catalog.pg_cast where castsource = 'int4'::regtype and casttarget = 'bool'::regtype;
oid | castsource | casttarget | castfunc | castcontext | castmethod
-----+-----+-----+-----+-----+-----
11311 | 23 | 16 | 2557 | e | f
(1 row)
```

The conversion is supported, and the conversion type is implicit conversion.

If no built-in conversion functions are available, customize a conversion function to support the conversion. For details, see [User-Defined Data Type Conversion](#).

User-Defined Data Type Conversion

- Use double colons (::) to perform a forcible conversion.

```
select '10'::int,'2023-10-05'::date;
int4 | date
-----+-----
10 | 2023-10-05
(1 row)
```

- Use the CAST function to convert the type.

```
select CAST('10' as int),CAST('2023-10-05' as date);
int4 | date
-----+-----
10 | 2023-10-05
(1 row)
```

- Customize a data type conversion.

For details, see <https://www.postgresql.org/docs/14/sql-createcast.html>.

NOTICE

Adding a custom type conversion will affect the existing execution plans of RDS for PostgreSQL. Therefore, customizing type conversions are not recommended.

- Conversion between time and character types
CREATE CAST(varchar as date) WITH INOUT AS IMPLICIT;
- Conversion between boolean types and numeric types
create cast(boolean as numeric) with INOUT AS IMPLICIT;
- Conversion between numeric types and character types
create cast(varchar as numeric) with INOUT AS IMPLICIT;

Example: Convert **text** to **date**.

```
create or replace function public.text_to_date(text) returns date as
$$
select to_date($1,'yyyy-mm-dd');
$$
language sql strict;

create cast (text as date) with function public.text_to_date(text) as implicit;

select text '2023-09-09' + 1;
?column?
-----
2023-09-10
(1 row)
```

3.4 Using Client Drivers to Implement Failover and Read/Write Splitting

Since PostgreSQL 10 (libpq.so.5.10), libpq has been supporting failover and read/write splitting, and Java Database Connectivity (JDBC) has been supporting read/write splitting, failover, and load balancing.

PostgreSQL client drivers are backward compatible. Even RDS for PostgreSQL 9.5 and 9.6 instances can be connected through the libpq driver of the latest version to implement failover.

NOTE

In this section, failover refers to the failover of read-only workloads.

- libpq is a C application programming interface (API) to PostgreSQL. libpq is a set of library functions that allow client programs to pass queries to the PostgreSQL backend server and to receive the results of these queries.
- JDBC is an API used in Java to define how client programs access databases. In PostgreSQL, JDBC supports failover and load balancing.

Table 3-3 Functions supported by libpq and JDBC

Driver	Read/Write Splitting	Load Balancing	Failover
libpq	√	×	√
JDBC	√	√	√

Using libpq for Failover and Read/Write Splitting

You can use libpq functions to connect to multiple databases. If one database fails, workloads are automatically switched to another available database.

postgresql://[user[:password]@][netloc][:port][,...][/dbname][?param1=value1&...]

Example: Connect to one primary RDS for PostgreSQL instance and two read replicas. Read requests will not fail as long as there is at least one available instance.

postgres://
<instance_ip>:*<instance_port>*,*<instance_ip>*:*<instance_port>*,*<instance_ip>*:*<instance_port>*|*<database_name>*?**target_session_attrs=any**

Table 3-4 Parameter description

Parameter	Description	Example Value
<i><instance_ip></i>	IP address of the DB instance.	If you attempt to access the instance from an ECS, set <i>instance_ip</i> to the floating IP address displayed on the Overview page of the instance. If you attempt to access the instance through an EIP, set <i>instance_ip</i> to the EIP that has been bound to the instance.
<i><instance_port></i>	Database port of the DB instance.	Set this parameter to the database port displayed on the Overview page. Default value: 5432
<i><database_name></i>	Name of the database to be connected.	The default management database is postgres . You can enter the database name based on the site requirements.

Parameter	Description	Example Value
target_session_attrs	Type of the database to be connected.	<ul style="list-style-type: none"> • any (default): libpq can connect to any database. If the connection is interrupted due to a fault in the database, libpq will attempt to connect to another database to implement failover. • read-write: libpq can only connect to a database that supports both read and write. libpq attempts a connection to the first database you specified. If this database supports only read or write operations, libpq disconnects from it and attempts to connect to the second one and so on until it connects to a database that supports both read and write. • read-only: libpq can only connect to a read-only database. libpq attempts a connection to the first database you specified. If this database is not a read-only database, libpq disconnects from it and attempts to connect to the second one and so on until it connects to a read-only database. This value is not supported in RDS for PostgreSQL 13 (libpq.so.5.13) or earlier versions.

For details about libpq and related parameters, see [Connection Strings](#).

You can use the `pg_is_in_recovery()` function in your application to determine whether the connected database is a primary instance (indicated by `f`) or a read replica to implement read/write splitting.

The following is an example of Python code (psycopg2 a wrapper for libpq):

```
// There will be security risks if the username and password used for authentication are directly written into code. Store the username and password in ciphertext in the configuration file or environment variables.
// In this example, the username and password are stored in the environment variables. Before running this example, set environment variables EXAMPLE_USERNAME_ENV and EXAMPLE_PASSWORD_ENV as needed.

import psycopg2
import os

username = os.getenv("EXAMPLE_USERNAME_ENV")
password = os.getenv("EXAMPLE_PASSWORD_ENV")
conn = psycopg2.connect(database=<database_name>,host=<instance_ip>, user=username,
password=password, port=<instance_port>, target_session_attrs="read-write")
cur = conn.cursor()
cur.execute("select pg_is_in_recovery()")
row = cur.fetchone()
print("recovery =", row[0])
```

Using JDBC for Failover and Read/Write Splitting

You can define multiple databases (hosts and ports) in the connection URL and separate them with commas (,). JDBC will attempt to connect to them in sequence

until the connection is successful. If the connection fails, an error message is displayed.

```
jdbc:postgresql://node1,node2,node3/${database}?  
targetServerType=preferSecondary&loadBalanceHosts=true
```

Example:

```
jdbc:postgresql://  
<instance_ip>:<instance_port>,<instance_ip>:<instance_port>,<instance_ip>:<instance_port>/<database_name>?  
targetServerType=preferSecondary&loadBalanceHosts=true
```

For details about the Java code, see [Connecting to an RDS for PostgreSQL Instance Through JDBC](#).

Table 3-5 Parameter description

Parameter	Description	Example Value
targetServerType	Type of the database to be connected.	<ul style="list-style-type: none"> • any: any database. • primary: primary database (writable and readable). For versions earlier than JDBC 42.2.0, use the parameter value master. • secondary: secondary database (readable). For versions earlier than JDBC 42.2.0, use the parameter value slave. • preferSecondary: The secondary database is preferred. If no secondary database is available, the primary database is connected. For versions earlier than JDBC 42.2.0, use the parameter value preferSlave.
loadBalanceHosts	Sequence of databases to be connected.	<ul style="list-style-type: none"> • False (default): Databases are connected in the sequence defined in the URL. • True: Databases are randomly connected.

 **NOTE**

To distinguish between the primary and secondary databases, check whether data can be written to the database. If yes, it is a primary database. If no, it is a secondary database. You can use the `pg_is_in_recovery()` function to determine whether a database is a primary database. For details, see [Using libpq for Failover and Read/Write Splitting](#).

To implement read/write splitting, you need to configure two data sources. For the first data source, set **targetServerType** to **primary** to process write requests. For the second data source:

- If there is only one read replica, set **targetServerType** to **preferSecondary** to process read requests. Assume that the IP addresses of the primary instance and read replica are 10.1.1.1 and 10.1.1.2, respectively.

```
jdbc:postgresql://10.1.1.2:5432,10.1.1.1:5432/${database}?  
targetServerType=preferSecondary
```

- If there are two read replicas, set **targetServerType** to **any** to process read requests. Assume that the IP addresses of the read replicas are 10.1.1.2 and 10.1.1.3, respectively.

```
jdbc:postgresql://10.1.1.2:5432,10.1.1.3:5432/${database}?  
targetServerType=any&loadBalanceHosts=true
```

3.5 Using PoWA

3.5.1 Overview

PoWA is an open-source system used to monitor the performance of RDS for PostgreSQL databases. It consists of the PoWA-archivist, PoWA-collector, and PoWA-web components and obtains performance data through other plug-ins installed in the RDS for PostgreSQL databases. The key components are as follows:

- PoWA-archivist: the PostgreSQL plug-in for collecting performance data obtained by other plug-ins.
- PoWA-collector: the daemon that gathers performance metrics from remote PostgreSQL instances on a dedicated repository server.
- PoWA-web: the web-based user interface displaying performance metrics collected by the PoWA-collector.
- Other plug-ins: the sources of performance metric data. They are installed on the destination PostgreSQL database.
- PoWA: the system name.

Security Risk Warning

The following security risks may exist during PoWA deployment and configuration.

- (Remote mode) When configuring instance performance metric information to be collected in powa-repository, you need to enter the IP address, **root** username, and connection password of the destination instance. You can query related information in the **powa_servers** table. The connection password is displayed in plaintext.
- In the **PoWA-collector** configuration file, the powa-repository connection information does not contain the connection password. It means that the powa-repository connection configuration item for PoWA-collector must be trust.
- In the **PoWA-web** configuration file, the **root** user and connection password of powa-repository (remote mode) or DB instance (local mode) are optional and stored in plaintext.

Before using the PoWA, you need to be aware of the preceding security risks. For details about how to harden security, see [PoWA official document](#).

3.5.2 Performance Metrics

Using PoWA has a minor negative impact on PostgreSQL server performance, and it is difficult to accurately assess this impact because it may come from different parts.

You need to activate at least the `pg_stat_statements` plug-in and other supported Stats plug-ins (if any). These plug-ins may slow down your instances and you are advised to choose an appropriate method to configure them.

If you do not use the remote mode, the data is periodically stored locally. Overhead varies with snapshot frequency and the disk usage affects backups.

3.5.2.1 Database Performance Metrics

General Overview

Figure 3-22 General Overview

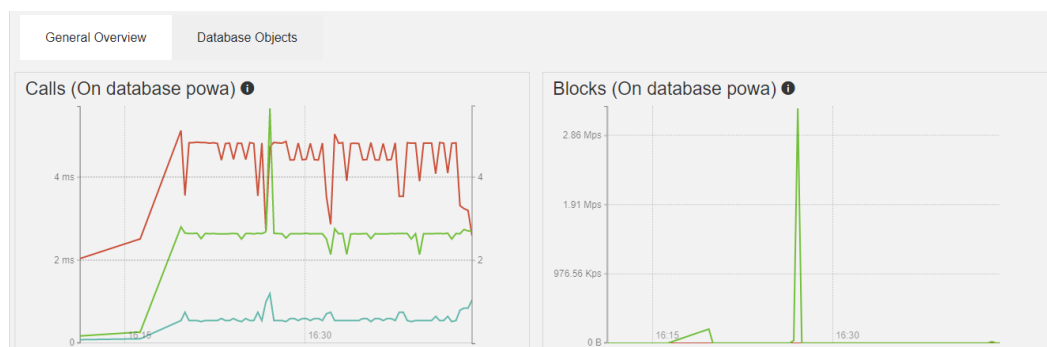


Table 3-6 Calls field description

Field	Description
Queries per sec	Number of queries executed per second
Runtime per sec	Total duration of queries executed per second
Avg runtime	Average query duration

Table 3-7 Blocks field description

Field	Description
Total shared buffers hit	Amount of data found in shared buffers
Total shared buffers miss	Amount of data found in OS cache or read from disk

Database Objects

Figure 3-23 Database Objects

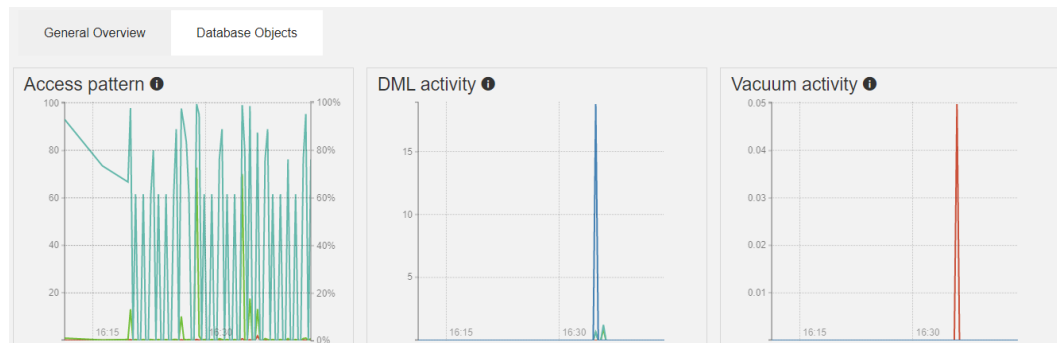


Table 3-8 Access pattern field description

Field	Description
Index scans ratio	Ratio of index scans to sequential scans
Index scans	Number of index scans per second
Sequential scans	Number of sequential scans per second

Table 3-9 DML activity field description

Field	Description
Tuples inserted	Number of tuples inserted per second
Tuples updated	Number of tuples updated per second
Tuples HOT updated	Number of heap-only tuples (HOT) updated per second
Tuples deleted	Number of tuples deleted per second

Table 3-10 Vacuum activity field description

Field	Description
# Vacuum	Number of vacuums per second
# Autovacuum	Number of autovacuum per second
# Analyze	Number of analyses per second
# Autoanalyze	Number of autoanalyses per second

Details for all databases

Figure 3-24 Details for all databases

Query	Execution			I/O Time			Blocks			Temp blocks	
	#	Time	Avg time	Read	Write	Read	Hit	Dirtyed	Written	Read	Written
COPY (SELECT 1, * FROM public.powa_user_functions_src(0)) TO stdout	75	3 s 783 ms	50 ms 442 µs	0	0	0 B	768.00 K	0 B	0 B	0 B	0 B
COPY (SELECT 1, * FROM public.powa_all_relations_src(0)) TO stdout	75	82 ms 323 µs	1 ms 98 µs	0	0	0 B	720.00 K	0 B	0 B	0 B	0 B
COPY (SELECT 1, * FROM public.powa_statements_src(0)) TO stdout	75	66 ms 99 µs	881 µs	0	0	0 B	5.38 M	0 B	0 B	0 B	0 B
select control_extension(\$1, \$2)	1	52 ms 437 µs	52 ms 437 µs	0	0	0 B	44.88 M	568.00 K	96.00 K	0 B	0 B
COPY (SELECT 1, * FROM public.powa_stat_bgwriter_src(0)) TO stdout	75	11 ms 886 µs	158 µs	0	0	0 B	688.00 K	0 B	0 B	0 B	0 B
COPY (SELECT 1, * FROM public.powa_databases_src(0)) TO stdout	75	11 ms 186 µs	149 µs	0	0	0 B	1.01 M	0 B	0 B	0 B	0 B
/* sql from das */select r.* from (SELECT n.nspname AS schema_name, c...	1	5 ms 355 µs	5 ms 355 µs	0	0	0 B	12.34 M	72.00 K	0 B	0 B	0 B
COPY (SELECT 1, * FROM public.pg_track_settings_settings_src(0)) TO st...	2	2 ms 367 µs	1 ms 184 µs	0	0	0 B	0 B	0 B	0 B	0 B	0 B
/* sql from das */select count(\$1) as table_count from information_sch...	1	916 µs	916 µs	0	0	0 B	7.29 M	8.00 K	0 B	0 B	0 B
SELECT pg_catalog.set_config(name, \$1, \$2) FROM pg_catalog.pg_settings...	1	783 µs	783 µs	0	0	0 B	0 B	0 B	0 B	0 B	0 B
SELECT setting FROM pg_settings WHERE name = \$1 --WHERE name = 'server...	1	696 µs	696 µs	0	0	0 B	0 B	0 B	0 B	0 B	0 B
SAVEPOINT src	384	322 µs	1 µs	0	0	0 B	0 B	0 B	0 B	0 B	0 B
SELECT \$1	35	148 µs	4 µs	0	0	0 B	0 B	0 B	0 B	0 B	0 B
/* sql from das */SELECT \$3 FROM pg_class c LEFT JOIN pg_namespace n 0...	1	148 µs	148 µs	0	0	0 B	136.00 K	0 B	0 B	0 B	0 B
COPY (SELECT 1, * FROM public.pg_track_settings_rds_src(0)) TO stdout	2	144 µs	72 µs	0	0	0 B	0 B	0 B	0 B	0 B	0 B

Table 3-11 Details for all databases field description

Field	Description
Query	SQL statement to be executed
(Execution) #	Number of times that the SQL statement is executed
(Execution) Time	Total execution time of the SQL statement
(Execution) Avg time	Average time for executing the SQL statement
(I/O Time) Read	Read I/O wait time
(I/O Time) Write	Write I/O wait time
(Blocks) Read	Number of disk read pages
(Blocks) Hit	Number of hit pages in the shared buffer
(Blocks) Dirtyed	Number of dirty pages
(Blocks) Written	Number of disk write pages
(Temp blocks) Read	Number of disk temporary read pages
(Temp blocks) Write	Number of disk temporary write pages

3.5.2.2 Instance Performance Metrics

General Overview

Figure 3-25 General Overview metrics

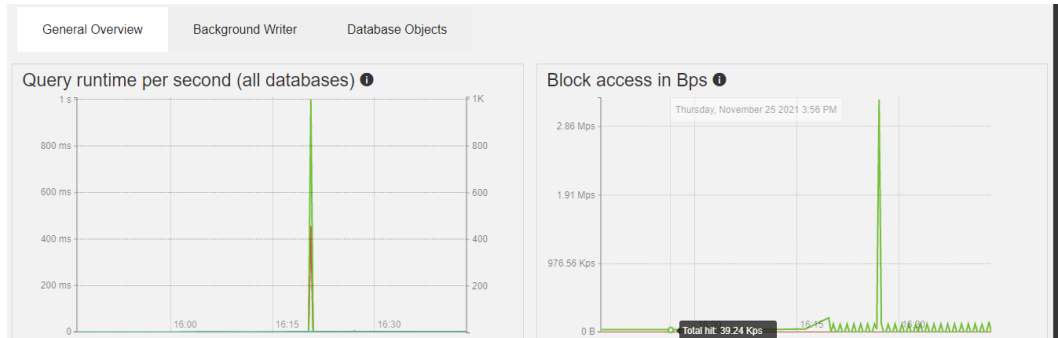


Table 3-12 Query runtime per second (all databases) field description

Field	Description
Queries per sec	Number of queries executed per second
Runtime per sec	Total duration of queries executed per second
Avg runtime	Average query duration

Table 3-13 Block access in Bps field description

Field	Description
Total hit	Amount of data found in shared buffers
Total read	Amount of data found in OS cache or read from disk

Background Writer

Figure 3-26 Background Writer metrics

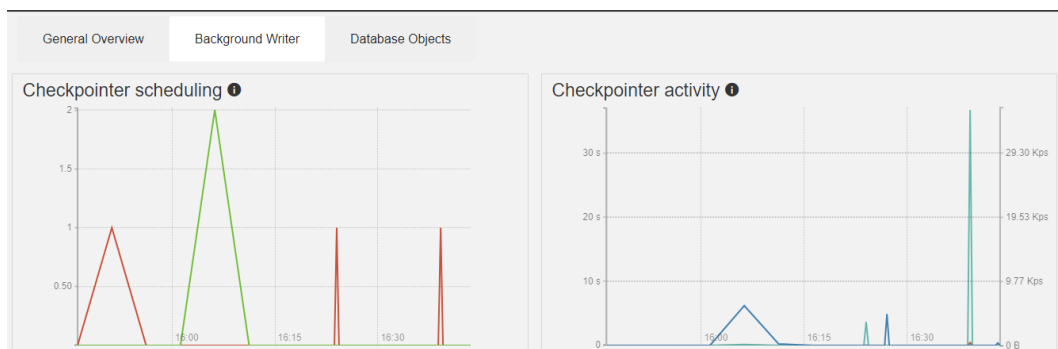


Table 3-14 Checkpointer scheduling field description

Field	Description
of requested checkpoints	Number of requested checkpoints that have been performed
of scheduled checkpoints	Number of scheduled checkpoints that have been performed

Table 3-15 Checkpointer activity field description

Field	Description
Buffers alloc	Number of buffers allocated
Sync time	Total amount of time that has been spent in the portion of checkpoint processing where files are synchronized to disk, in milliseconds
Write time	Total amount of time that has been spent in the portion of checkpoint processing where files are written to disk, in milliseconds

Table 3-16 Background writer field description

Field	Description
Maxwritten clean	Number of times the background writer stopped a cleaning scan because it had written too many buffers
Buffers clean	Number of buffers written by the background writer

Table 3-17 Backends field description

Field	Description
Buffers backend fsync	Number of times a backend had to execute its own fsync call (normally the background writer handles those even when the backend does its own write)
Buffers backend	Number of buffers written directly by a backend

Database Objects

Figure 3-27 Database Objects metrics

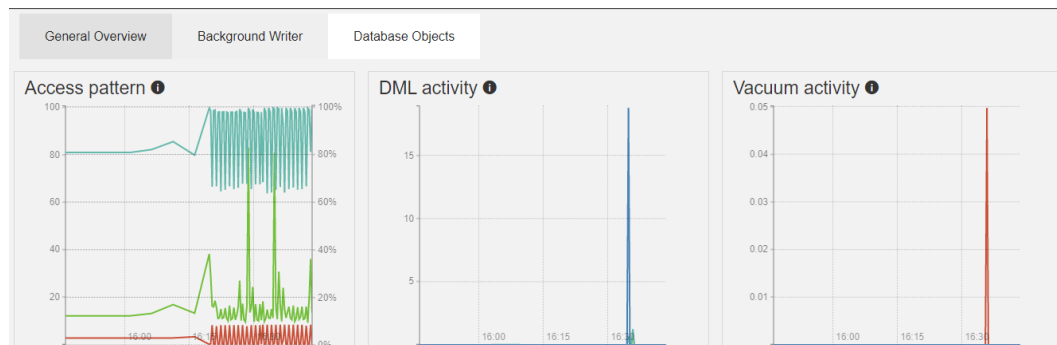


Table 3-18 Access pattern field description

Field	Description
Index scans ratio	Ratio of index scans to sequential scans
Index scans	Number of index scans per second
Sequential scans	Number of sequential scans per second

Table 3-19 DML activity field description

Field	Description
Tuples inserted	Number of tuples inserted per second
Tuples updated	Number of tuples updated per second
Tuples HOT updated	Number of heap-only tuples (HOT) updated per second
Tuples deleted	Number of tuples deleted per second

Table 3-20 Vacuum activity field description

Field	Description
# Vacuum	Number of vacuums per second
# Autovacuum	Number of autovacuum per second
# Analyze	Number of analyses per second
# Autoanalyze	Number of autoanalyses per second

Details for all databases

Figure 3-28 Details for all databases metrics

Database	#Calls	Runtime	Avg runtime	Blocks read	Blocks hit	Blocks dirtied	Blocks written	Temp Blocks written	I/O time
postgres	4,340	817 ms 136 µs	190 µs	0 B	133.86 M	56.00 K	0 B	0 B	0
powa	983	4 s 128 ms	4 ms 200 µs	8.00 K	75.25 M	664.00 K	96.00 K	0 B	20 µs
test	238	20 s 18 ms	84 ms 110 µs	8.00 K	864.00 K	0 B	0 B	0 B	10 µs

Table 3-21

Field	Description
Database	Database name
#Calls	Total number of executed SQL statements
Runtime	Total runtime of the SQL statement
Avg runtime	Average runtime of the SQL statement
Blocks read	Number of pages read from the disk
Blocks hit	Number of hit pages in the shared buffer
Blocks dirtied	Number of dirty pages
Blocks written	Number of disk write pages
Temp Blocks written	Number of disk temporary write pages
I/O time	I/O wait time

3.5.3 PoWA Deployment Models

PoWA supports local and remote deployment. For details, see [Local Deployment](#) and [Remote Deployment](#). For security purposes, RDS for PostgreSQL supports only remote deployment of PoWA.

Local Deployment

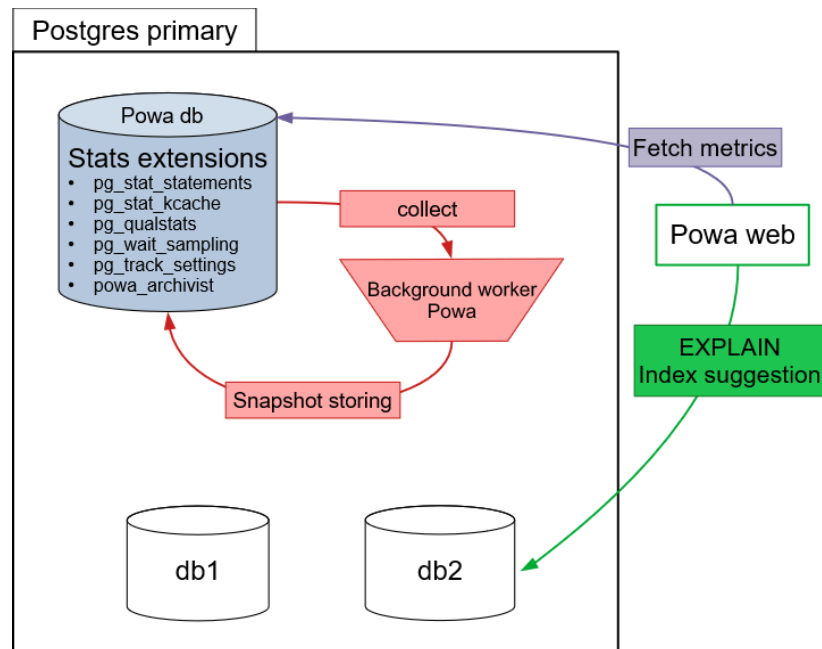
The architecture is simple and does not require the participation of the PoWA-collector. It consists of only the PoWA-archivist, PoWA-web, and other plug-ins. However, this architecture has the following disadvantages:

- It increases performance costs when collecting data and using the user interface.
- Collected performance metric data is stored on the local host, which increases the disk space usage.
- Data cannot be collected on the hot standby server.

NOTICE

For security purposes, RDS for PostgreSQL does not support local deployment.

Figure 3-29 Local deployment architecture



Remote Deployment

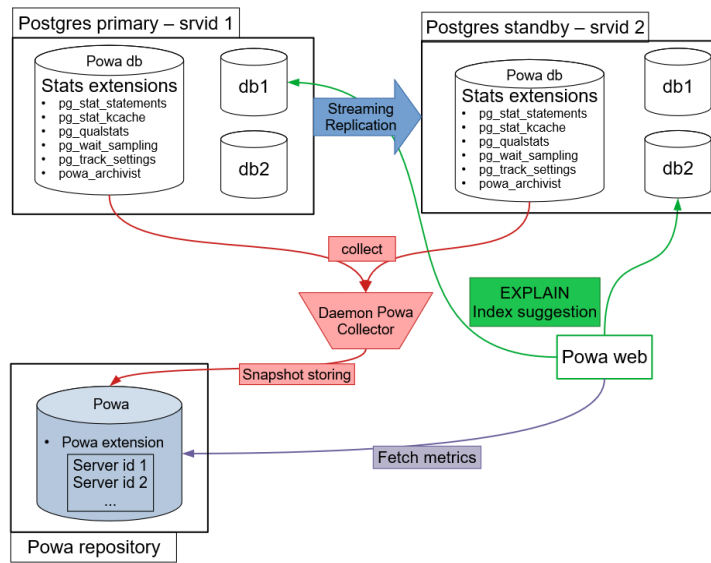
PoWA 4.0 allows you to store the data of one or multiples servers on an external PostgreSQL database. Currently, RDS for PostgreSQL supports PoWA 4.1.2.

Compared with local deployment, remote deployment has a dedicated **Powa repository** database for storing performance metric data collected by PoWA-collector.

RDS for PostgreSQL supports remote deployment.

The remote deployment architecture is as follows.

Figure 3-30 Remote deployment architecture



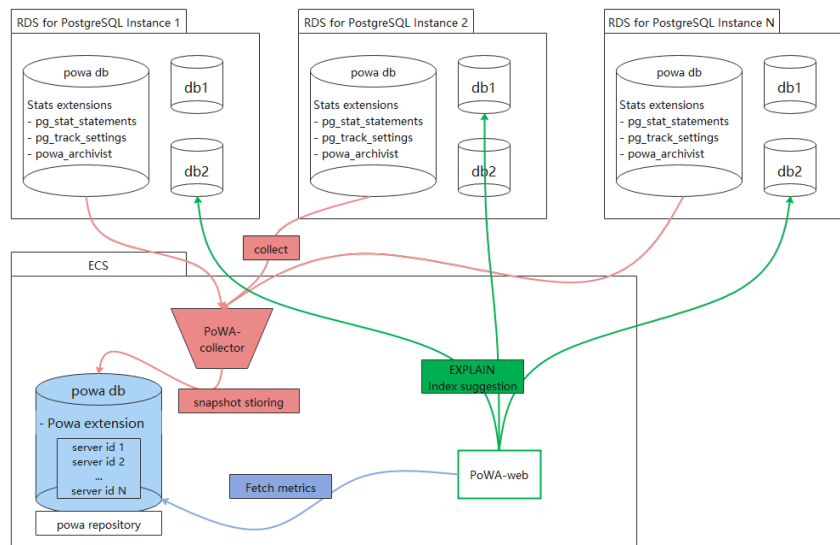
3.5.3.1 Deploying PoWA for an RDS for PostgreSQL Instance

To remotely deploy PoWA on Huawei Cloud, There must be an ECS with PoWA-archivist, PoWA-collector, and PoWA-web installed on it. This section describes how to install PoWA-archivist, PoWA-collector, and PoWA-web.

Architecture

The remote deployment architecture is as follows:

Figure 3-31 Remote deployment architecture



Preparations

- An RDS for PostgreSQL 12.6 instance has been created.

- An ECS has been created and bound with an EIP. In this example, the ECS uses the CentOS 8.2 64-bit image.

Installing Python3

PoWA-collector and PoWA-web must be installed in a Python3 environment. You can use pip3 to install them to facilitate the installation. In this example, Python 3.6.8 is installed on the ECS by default. The latest PoWA version fails to be installed. For details about how to install the latest version, see [Installing Python 3.9.9](#).

Installing PoWA-archivist

1. Run the **wget** command to obtain the PoWA-archivist source code.

```
wget https://github.com/powa-team/powa-archivist/archive/refs/tags/REL_4_1_2.tar.gz
```
2. Decompress the downloaded **REL_4_1_2.tar.gz** package.
3. Install PoWA-archivist to the decompressed directory.

```
make && make install
```

Installing PoWA-collector and PoWA-web

1. Switch to the RDS for PostgreSQL database user. Take user **postgres** as an example.

```
su - postgres
```
2. Install PoWA-collector and PoWA-web. **psycopg2** is mandatory for installing them.

```
pip install psycopg2  
pip install powa-collector  
pip install powa-web
```

After the installation is complete, check the following path tree. If the following information is displayed, the PoWA-collector and PoWA-web have been installed.

```
/home/postgres/.local/bin  
├── powa-collector.py  
├── powa-web  
└── __pycache__
```

Creating the PoWA Extension

Step 1 Log in to the powa database of the RDS for PostgreSQL instance as user **root**. (If the powa database does not exist, create it first.)

Step 2 Create the powa extension in the powa database.

```
select control_extension('create', 'pg_stat_statements');  
select control_extension('create', 'btree_gist');  
select control_extension('create', 'powa');
```

----End

FAQs

Q: What should I do if the error message "python setup.py build_ext --pg-config / path/to/pg_config build" is displayed when the **pip install psycopg2** command is executed?

A: You need to add the **bin** and **lib** paths of RDS for PostgreSQL to environment variables and run the **pip install psycopg2** command.

Installing Python 3.9.9

1. Prepare the environment.

Perform the following operations in sequence. Otherwise, Python 3.9.9 may fail to be installed (for example, SSL component dependency fails). As a result, PoWA-collector and PoWA-web fail to be installed.

```
yum install readline* -y
yum install zlib* -y
yum install gcc-c++ -y
yum install sqlite* -y
yum install openssl* -y
yum install libffi* -y
```

2. Install Python 3.9.9.

- a. Run the following commands as user **root**:

```
mkdir env
cd env
wget https://www.python.org/ftp/python/3.9.9/Python-3.9.9.tgz
tar -xvzf Python-3.9.9.tgz
cd Python-3.9.9
./configure --prefix=/usr/local/python3.9.9
make && make install
```

- b. Create a soft link.

```
ln -s /usr/local/python3.9.9/bin/python3.9 /usr/bin/python
ln -s /usr/local/python3.9.9/bin/pip3.9 /usr/bin/pip
```

3. Check whether the installation is successful.

- a. Verify the installation, especially the SSL function.

```
[root@ecs-ad4d Python-3.9.9]# python
Python 3.9.9 (main, Nov 25 2021, 12:36:32)
[GCC 8.4.1 20200928 (Red Hat 8.4.1-1)] on linux
Type "help", "copyright", "credits" or "license" for more information.
import ssl
import urllib.request
context = ssl._create_unverified_context()
urllib.request.urlopen('https://www.example.com/',context=context).read()
```

- b. If any command output is displayed, the installation is successful. Run the following command to exit:

```
quit()
```

3.5.3.2 Configuring Remote Deployment

Configuring a DB Instance for Performance Metric Collection

Step 1 Log in to the powa database of the target DB instance as user **root**. (If the powa database does not exist, create it first.)

Step 2 Create the powa plug-in in the powa database.

```
select control_extension('create', 'pg_stat_statements');
select control_extension('create', 'btree_gist');
select control_extension('create', 'powa');
```

----End

Configuring Local PostgreSQL

Take the PostgreSQL (powa-repository) on the ECS as an example.

- Version: PostgreSQL 12.6
- superuser: postgres
- Data path: `/home/postgres/data`

Step 1 Add `pg_stat_statements` to `shared_preload_libraries` in the `/home/postgres/data/postgresql.conf` file.

```
shared_preload_libraries = 'pg_stat_statements' # (change requires restart)
```

Step 2 Restart the database.

```
pg_ctl restart -D /home/postgres/data/
```

Step 3 Log in to the database as the **super** user, create database **powa**, and install related plug-ins.

NOTICE

The created database must be named **powa**. Otherwise, an error is reported and certain functions do not take effect while PoWA is running.

```
[postgres@ecs-ad4d ~]$ psql -U postgres -d postgres
psql (12.6)
Type "help" for help.
postgres=# create database powa;
CREATE DATABASE
postgres=# \c powa
You are now connected to database "powa" as user "postgres".
powa=# create extension pg_stat_statements ;
CREATE EXTENSION
powa=# create extension btree_gist ;
CREATE EXTENSION
powa=# create extension powa;
CREATE EXTENSION
```

Step 4 Configure the instance whose performance metrics need to be collected.

1. Add the instance information.

```
powa=# select powa_register_server(
  hostname => '192.168.0.1',
  alias => 'myInstance',
  port => 5432,
  username => 'user1',
  password => '*****',
  frequency => 300);
 powa_register_server
-----
 t
(1 row)
```

2. View the **powa_servers** table to obtain the performance metric information of the instance.

```
powa=# select * from powa_servers;
 id | hostname | alias | port | username | password | dbname | frequency | powa_coalesce | retention |
 allow_ui_connection | version
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
 0 |          | <local> | 0 |          |          |          | -1 | 100 | 00:00:00 | t |          |
```

```
1 | 192.168.0.1 | myInstance | user1 | 5432 | ***** | powa | 300 | 100 | 1 day |  
t |  
(2 rows)
```

NOTICE

Information about the destination instance information includes important privacy information, such as its IP address, root user account, and plaintext password.

Assess the plug-in security risks before you decide to use them.

----End

Configuring PoWA-collector

When PoWA-collector starts, it searches for configuration files in the following sequence:

1. /etc/powa-collector.conf
2. ~/.config/powa-collector.conf
3. ~/.powa-collector.conf
4. ./powa-collector.conf

The configuration files must contain the following options:

- repository.dsn: URL. It is used to notify the powa-collector of how to connect to the dedicated storage database (powa-repository).
- debug: Boolean type. It specifies whether to enable the powa-collector in debugging mode.

Take the configuration file **./powa-collector.conf** as an example.

```
{  
  "repository": {  
    "dsn": "postgresql://postgres@localhost:5432/powa"  
  },  
  "debug": true  
}
```

No password is configured in the PoWA-collector configuration. Therefore, you need to set the connection policy in the **pg_hba.conf** file of the **powa-repository** database to **trust** (password-free connection).

Start the powa-collector.

```
cd /home/postgres/.local/bin  
./powa-collector.py &
```

Configuring PoWA-Web

When PoWA-collector starts, it searches for configuration files in the following sequence:

1. /etc/powa-web.conf

2. `~/config/powa-web.conf`
3. `~/powa-web.conf`
4. `./powa-web.conf`

Take the configuration file `./powa-web.conf` as an example.

```
# cd /home/postgres/.local/bin
# vim ./powa-web.conf
# Write the configuration information and save it.
servers={
  'main': {
    'host': 'localhost',
    'port': '5432',
    'database': 'powa',
    'username': 'postgres',
    'query': {'client_encoding': 'utf8'}
  }
}
cookie_secret="SECRET_STRING"
```

In this example, the connection policy in the `pg_hba.conf` file of the powa-repository database is set to **trust** (password-free connection). Therefore, the password is not configured.

Start the powa-web.

```
cd /home/postgres/.local/bin
```

```
./powa-web &
```

3.5.4 Accessing PoWA

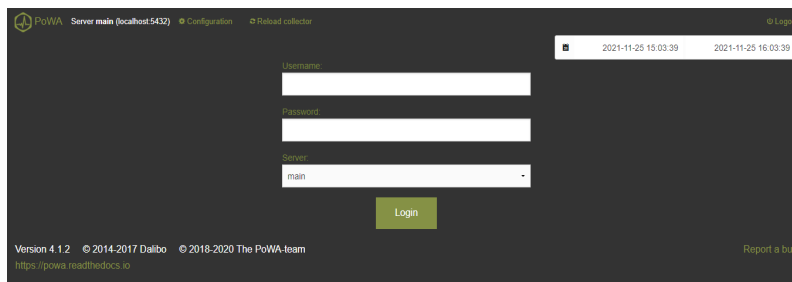
After the remote deployment is configured, and PoWA-collector and PoWA-web are started, you can view the monitoring metrics of the instance through a browser. In the example,

Port is not configured in the `powa-web.conf` file. The default value **8888** is used.

Browser access website: http://ECS_IP_address:8888/

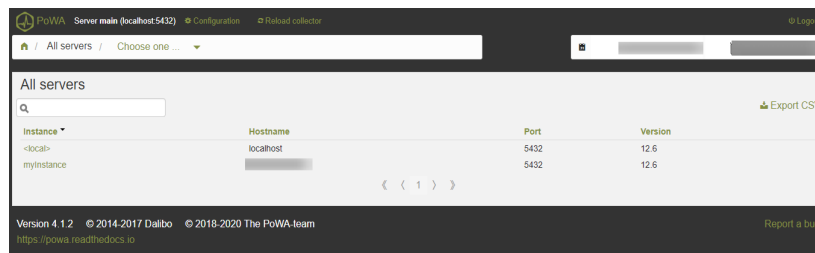
Step 1 Use a browser to access PoWA.

Figure 3-32 Access PoWA.



Step 2 Enter the username and password, and click **Login**.

Figure 3-33 PoWA home page



The PoWA collects information about two PostgreSQL instances.

- **<local>**: PostgreSQL database built on the ECS, which is used as the powa-repository role.
- **myinstance**: RDS for PostgreSQL instance, which is used as the target for performance data collection. (**myinstance** is the instance alias registered in powa-repository.)

Step 3 Click an instance to view its performance metrics.

----End

3.6 Best Practices for Using pg_dump

Description

pg_dump is a native tool for backing up a PostgreSQL database. The file created by pg_dump can be a SQL script file or an archive file. For details, see [pg_dump](#).

- **SQL script file**: It is a plain-text file that contains the SQL commands required to rebuild a database to the state when it was backed up.
- **Archive file**: It must be used with pg_restore to rebuild a database. This format allows pg_restore to select the data to be restored.

Precautions

pg_dump dumps a single database, schemas, or tables. Only tables, data, and functions can be exported. Before restoration, you need to create a database and account in the target instance in advance.

- **--format=custom**: The dump file is in binary format, which can be used only by pg_restore. You can restore specific tables from a dump file.
- **--format=plain**: The dump file is in plain-text format. To restore from such a plain-text file, connect to the database and execute the file.

Constraints

Before using pg_dump and pg_restore, ensure that the versions of the source and target databases are the same to avoid compatibility issues. Incompatible versions may cause data loss or restoration errors.

Preparing Test Data

```
# Create a database.
create database dump_database;

# Log in to the database.
\c dump_database

# Create table 1 and insert data into the table.
create table dump_table(id int primary key, content char(50));
insert into dump_table values(1,'aa');
insert into dump_table values(2,'bb');

# Create table 2 and insert data into the table.
create table dump_table2(id int primary key, content char(50));
insert into dump_table2 values(1,'aaaa');
insert into dump_table2 values(2,'bbbb');
```

Using pg_dump to Export a Database to a SQL File

Syntax

```
pg_dump --username=<DB_USER> --host=<DB_IPADDRESS> --port=<DB_PORT> --format=plain --
file=<BACKUP_FILE><DB_NAME>
```

- *DB_USER* indicates the database username.
- *DB_IPADDRESS* indicates the database address.
- *DB_PORT* indicates the database port.
- *BACKUP_FILE* indicates the name of the file to be exported.
- *DB_NAME* indicates the name of the database to be exported.
- **--format** indicates the format of the exported file. **plain** (default) indicates a plain-text file that contains SQL scripts. For details about other options, see [pg_dump](#).

Examples

- Export a database to a SQL file (INSERT statements).

```
$ pg_dump --username=root --host=192.168.61.143 --port=5432 --format=plain --inserts --
file=backup.sql dump_database
```

 Password for user root:
- Export all table schemas from a database to a SQL file.

```
$ pg_dump --username=root --host=192.168.61.143 --port=5432 --format=plain --schema-only --
file=backup.sql dump_database
```

 Password for user root:
- Export all table data from a database to a SQL file.

```
$ pg_dump --username=root --host=192.168.61.143 --port=5432 --format=plain --data-only --
file=backup.sql dump_database
```

 Password for user root:

After the commands in any of the above examples are executed, a **backup.sql** file will be generated as follows:

```
[rds@localhost ~]$ ll backup.sql
-rw-r----- 1 rds rds 5657 May 24 09:21 backup.sql
```

Using pg_dump to Export Specified Tables from a Database to a SQL File

Syntax

```
pg_dump --username=<DB_USER> --host=<DB_ADDRESS> --port=<DB_PORT> --format=plain --file=<BACKUP_FILE> <DB_NAME> --table=<TABLE_NAME>
```

- *DB_USER* indicates the database username.
- *DB_ADDRESS* indicates the database address.
- *DB_PORT* indicates the database port.
- *BACKUP_FILE* indicates the name of the file to be exported.
- **DB_NAME** indicates the name of the database to be migrated.
- *TABLE_NAME* indicates the name of the specified table in the database to be migrated.
- **--format** indicates the format of the exported file. **plain** (default) indicates a plain-text file that contains SQL scripts. For details about other options, see [pg_dump](#).

Examples

- Export a single table from a database to a SQL file.

```
$ pg_dump --username=root --host=192.168.61.143 --port=5432 --format=plain --file=backup.sql dump_database --table=dump_table
```

 Password for user root
- Export multiple tables from a database to a SQL file.

```
$ pg_dump --username=root --host=192.168.61.143 --port=5432 --format=plain --file=backup.sql dump_database --table=dump_table --table=dump_table2
```

 Password for user root:
- Export all tables starting with **ts_** from a database to a SQL file.

```
$ pg_dump --username=root --host=192.168.61.143 --port=5432 --format=plain --file=backup.sql dump_database --table=ts_*
```

 Password for user root:
- Export all tables excluding those starting with **ts_** from a database to a SQL file.

```
$ pg_dump --username=root --host=192.168.61.143 --port=5432 --format=plain --file=backup.sql dump_database -T=ts_*
```

 Password for user root:

After the commands in any of the above examples are executed, a **backup.sql** file will be generated as follows:

```
[rds@localhost ~]$ ll backup.sql
-rw-r----- 1 rds rds 5657 May 24 09:21 backup.sql
```

Using pg_dump to Export Data of a Specific Schema

Syntax

```
pg_dump --username=<DB_USER> --host=<DB_IPADDRESS> --port=<DB_PORT> --format=plain --schema=<SCHEMA> <DB_NAME> --table=<BACKUP_FILE>
```

- *DB_USER* indicates the database username.
- *DB_IPADDRESS* indicates the database address.
- *DB_PORT* indicates the database port.
- *BACKUP_FILE* indicates the name of the file to be exported.
- *DB_NAME* indicates the name of the database to be exported.
- *SCHEMA* indicates the name of the schema to be exported.
- **--format** indicates the format of the exported file. **plain** (default) indicates a plain-text file that contains SQL scripts. For details about other options, see [pg_dump](#).

Examples

- Export all data of the public schema from a database.

```
pg_dump --username=root --host=192.168.61.143 --port=5432 --format=plain --schema=public  
dump_database > backup.sql
```
- Export all data except the public schema from a database in a customized compression format.

```
pg_dump --username=root --host=192.168.61.143 --port=5432 --format=custom -b -v -N public  
dump_database > all_sch_except_pub.backup
```

Restoring Data

To restore from a plain-text SQL script file, run the **psql** command. See the following example commands:

```
# Restore a specific database.  
psql --username=root --host=192.168.61.143 --port=5432 backup_database < backup.sql  
  
# Restore a specific table.  
psql --username=root --host=192.168.61.143 --port=5432 backup_database --  
table=dump_table < backup.sql  
  
# Restore a specific schema.  
psql --username=root --host=192.168.61.143 --port=5432 backup_database --schema=public <  
backup.sql
```

NOTE

Before the restoration, create a database named **backup_database** in the target database.

To restore from other file formats, use **pg_restore**. **pg_restore** is used to restore a PostgreSQL database in any non-plain-text format dumped by **pg_dump**.

```
pg_restore --username=root --host=192.168.61.143 --port=5432 --dbname=backup_database --  
format=custom all_sch_except_pub.backup --verbose
```

FAQ

1. What should I do if an error about insufficient permissions is reported for **pg_dump**?

Solution:

Check whether the **root** user is used to export data. If any other user account is used, an error about insufficient permissions will be reported. If the **root** user is used and an error is still reported, check the database version. You can run **pg_dump** commands as the **root** user only when the kernel version support **root** privilege escalation. For details about the kernel versions that support **root** privilege escalation, see [Privileges of the root User](#).

2. Why an error was reported for functions such as **control_extension** after I imported a dump file to the target RDS for PostgreSQL database?

Solution:

That's because the target database contains these functions. This error can be ignored.

3.7 Best Practices for Using PgBouncer

Introduction to PgBouncer

PgBouncer is a lightweight connection pooler for PostgreSQL. It can:

- Cache connections to PostgreSQL. When a connection request is received, an idle process is allocated. PostgreSQL does not need to fork a new process to establish a connection. No resources need to be used for creating a new process and establishing a connection.
- Improve the connection usage and prevent excessive invalid connections from consuming too many database resources and causing high CPU usage.
- Restrict client connections to prevent excessive or malicious connection requests.

It is lightweight because:

- It uses libevent for socket communication, improving the communication efficiency.
- It uses C language and only 2 KB of memory is consumed by each connection.

PgBouncer supports the following types of connection pooling:

- Session pooling: PgBouncer does not reclaim the allocated connection until the client session ends.
- Transaction pooling: PgBouncer reclaims the allocated connection after the transaction is complete. The client only has exclusive access to a connection during a transaction. Non-transaction requests do not have exclusive connections.
- Statement pooling: PgBouncer reclaims the connection anytime a database request completes. In this mode, the client cannot use transactions. Using transactions in this case will cause data inconsistency.

The default pooling type for PgBouncer is session. You are advised to change it to transaction.

Installation and Configuration

Before deploying PgBouncer on the cloud, [purchase an ECS](#). To reduce network latency, you are advised to select the same VPC and subnet as those of the backend RDS instance for the ECS. After the purchase is complete, log in to the ECS to set up the environment.

1. PgBouncer is based on libevent, so you need to install the **libevent-devel** and **openssl-devel** dependencies.

```
yum install -y libevent-devel  
yum install -y openssl-devel
```

2. After that, download the source code from the PgBouncer official website and compile the code and install PgBouncer as a regular user.

```
su - pgbouncer  
tar -zxvf pgbouncer-1.19.0.tar.gz  
cd pgbouncer-1.19.0  
./configure --prefix=/usr/local
```

```
make
make install
```

3. Create the following directories to store the files (such as logs and process IDs) generated by PgBouncer:

```
mkdir -p /etc/pgbouncer/
mkdir -p /var/log/pgbouncer/
mkdir -p /var/run/pgbouncer/
```

4. Before starting PgBouncer, build the configuration file **pgbouncer.ini**.

```
[databases]
* = host=127.0.0.1 port=5432
[pgbouncer]
logfile = /var/log/pgbouncer/pgbouncer.log
pidfile = /var/run/pgbouncer/pgbouncer.pid
listen_addr = *
listen_port = 6432
auth_type = md5
auth_file = /etc/pgbouncer/userlist.txt
admin_users = postgres
stats_users = stats, postgres
pool_mode = transaction
server_reset_query = DISCARD ALL
max_client_conn = 100
default_pool_size = 20
;; resolve: unsupported startup parameter: extra_float_digits
;;ignore_startup_parameters = extra_float_digits
```

For details about the parameters in the configuration file, see the [official PgBouncer documentation](#).

Starting PgBouncer

PgBouncer cannot be started as **root**. It has to be started as a regular user.

```
pgbouncer -d /etc/pgbouncer/pgbouncer.ini
```

After it is started, run **netstat -tunlp | grep pgbouncer** to check the listening port of the connection pool and then connect to the DB instance.

```
psql -U root -d postgres -h 127.0.0.1 -p 6432
Password for user root:
psql (12.13)
Type "help" for help.
postgres=> \l

              List of databases
  Name      | Owner      | Encoding | Collate | Ctype  | Access privileges
-----+-----+-----+-----+-----+-----
 postgres  | pgbouncer  | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0 | pgbouncer  | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/pgbouncer
           |           |          |             |             | pgbouncer=CTc/pgbouncer
 template1 | pgbouncer  | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/pgbouncer
```

Stopping PgBouncer

You can run the **kill** command to stop it.

```
kill `cat /var/run/pgbouncer/pgbouncer.pid`
cat /var/run/pgbouncer/pgbouncer.pid | xargs kill -9
```

PgBouncer Management

PgBouncer provides a virtual database **pgbouncer**, which provides a database operation interface like PostgreSQL. It is not a real database, but a command line interface virtualized by PgBouncer. To log in to this virtual database, run the following command:

```
psql -p 6432 -d pgbouncer
```

If some configuration parameters are modified, you do not need to restart PgBouncer but run **reload** for the modifications to be applied.

```
pgbouncer=# reload;
RELOAD
```

After login, you can run **show help** to check the command help, run **show clients** to check the client connection information, and run **show pools** to check the connection pool information.

An Example for Read/Write Splitting

PgBouncer cannot automatically parse or split read and write requests. Read and write requests need to be distinguished on the application side.

1. Modify the database information in the **pgbouncer.ini** file and add the connection configurations of the primary instance and read replica to the file. In this example, the parameters are set as follows:

```
[databases]
;; * = host=127.0.0.1 port=5432
# The connection information of the read replica.
mydb_read: host=10.7.131.69 port=5432 dbname=postgres user=root password=***
# The connection information of the primary instance.
mydb_write: host=10.8.115.171 port=5432 dbname=postgres user=root password=***
[pgbouncer]
logfile = /var/log/pgbouncer/pgbouncer.log
pidfile = /var/run/pgbouncer/pgbouncer.pid
listen_addr = *
listen_port = 6432
auth_type = md5
auth_file = /etc/pgbouncer/userlist.txt
admin_users = postgres
stats_users = stats, postgres
pool_mode = transaction
server_reset_query = DISCARD ALL
max_client_conn = 100
default_pool_size = 20
;; resolve: unsupported startup parameter: extra_float_digits
;;ignore_startup_parameters = extra_float_digits
```

2. Check whether the primary instance and read replica can be connected. The primary instance and read replica have been connected using psql and read/write splitting is supported.

```
psql -U root -d mydb_write -h 127.0.0.1 -p 6432
Password for user root:
psql (14.6)
mydb_write=> SELECT pg_is_in_recovery();
 pg_is_in_recovery
-----
 f
(1 row)
psql -U root -d mydb_read -h 127.0.0.1 -p 6432
Password for user root:
psql (14.6)
mydb_read=> SELECT pg_is_in_recovery();
 pg_is_in_recovery
-----
 t
(1 row)
```

3.8 Security Best Practices

PostgreSQL has earned a reputation for reliability, stability, and data consistency, and has become the preferred choice as an open-source relational database for many enterprises. RDS for PostgreSQL is a cloud-based web service that is reliable, scalable, easy to manage, and immediately ready for use.

Make security configurations from the following dimensions to meet your service needs.

- [Configuring the Maximum Number of Connections to the Database](#)
- [Configuring the Timeout for Client Authentication](#)
- [Configuring SSL and Encryption Algorithm](#)
- [Configuring Password Encryption](#)
- [Disabling the Backslash Quote](#)
- [Periodically Checking and Deleting Roles That Are No Longer Used](#)
- [Revoking All Permissions on the public Schema](#)
- [Setting a Proper Password Validity Period for a User Role](#)
- [Configuring the Log Level to Record SQL Statements That Cause Errors](#)
- [Configuring Least-Privilege Permissions for Database Accounts](#)
- [Enabling Data Backup](#)
- [Enabling Database Audit](#)
- [Avoiding Binding an EIP to Your RDS for PostgreSQL Instance](#)
- [Updating the Database Version to the Latest](#)
- [Configuring the Delay for Account Authentication Failures](#)

Configuring the Maximum Number of Connections to the Database

The **max_connections** parameter specifies the maximum concurrent connections allowed in a database. If the value of this parameter is large, the RDS for PostgreSQL database may request more System V shared memory or semaphore. As a result, the requested shared memory or semaphore may exceed the default value on the OS. Set **max_connections** based on service complexity. For details, see [Instance Usage Suggestions](#).

Configuring the Timeout for Client Authentication

The **authentication_timeout** parameter specifies the maximum duration allowed to complete client authentication, in seconds. This parameter prevents clients from occupying a connection for a long time. The default value is 60s. If client authentication is not complete within the specified period, the connection is forcibly closed. Using this parameter can enhance the security of your RDS for PostgreSQL instance.

Configuring SSL and Encryption Algorithm

SSL is recommended for TCP/IP connections because SSL ensures that all communications between clients and servers are encrypted, preventing data

leakage and tampering and ensuring data integrity. When configuring SSL, configure the TLS protocol and encryption algorithm on the server. TLSv1.2 and ECDH+ECDSA+AESGCM:ECDH+aRSA+AESGCM:EDH+aRSA+AESGCM:EDH+aDSS+AESGCM:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!SRP:!RC4 are recommended. For details, see [SSL Connection](#).

To configure the TLS protocol and encryption algorithm, use the parameters `ssl_min_protocol_version` and `ssl_ciphers`.

Configuring Password Encryption

Passwords must be encrypted. When you use **CREATE USER** or **ALTER ROLE** to change a password, the password is stored in a system catalog after being encrypted by default. **scram-sha-256** is recommended for password encryption. To change the password encryption algorithm, change the value of **password_encryption**.

The **MD5** option is used only for compatibility with earlier versions. New DB instances use **scram-sha-256** by default.

NOTICE

The modification of **password_encryption** takes effect only after the password is reset.

Disabling the Backslash Quote

The **backslash_quote** parameter specifies whether a single quotation mark (') in a string can be replaced by a backslash quote (\'). The preferred, SQL-standard way to represent a single quotation mark is by doubling it ("). If client-side code does escaping incorrectly then an SQL-injection attack is possible. You are advised to set **backslash_quote** to **safe_encoding** to reject queries in which a single quotation mark appears to be escaped by a backslash, preventing SQL injection risks.

Periodically Checking and Deleting Roles That Are No Longer Used

Check whether all roles are mandatory. Every unknown role must be reviewed to ensure that it is used properly. If any role is no longer used, delete it. To query roles, run the following command:

```
SELECT rolname FROM pg_roles;
```

Revoking All Permissions on the public Schema

The **public** schema is the default schema. All users can access objects in it, including tables, functions, and views, which may cause security vulnerabilities. You can run the following command as user **root** to revoke the permissions:

```
revoke all on schema public from public;
```

Setting a Proper Password Validity Period for a User Role

When creating a role, you can use the **VALID UNTIL** keyword to specify when the password of the role becomes invalid. If this keyword is ignored, the password will

be valid permanently. You are advised to change the password periodically, for example, every three months. To configure a password validity period, run the following command:

```
CREATE ROLE name WITH PASSWORD 'password' VALID UNTIL 'timestamp';
```

To check whether a password validity period is configured, run the following command:

```
SELECT rolname,rolvaliduntil FROM pg\roles WHERE rolsuper = false AND rolvaliduntil IS NULL;
```

Configuring the Log Level to Record SQL Statements That Cause Errors

The `log_min_error_statement` parameter specifies which SQL statements that cause errors can be recorded in server logs. The SQL statements of the specified level or higher are recorded in logs. Valid values include **debug5**, **debug4**, **debug3**, **debug2**, **debug1**, **info**, **notice**, **warning**, **error**, **log**, **fatal**, and **panic**. The value of `log_min_error_statement` must be at least **error**. For details, see [Log Reporting](#).

Configuring Least-Privilege Permissions for Database Accounts

RDS for PostgreSQL allows you to grant role-based permissions to a database account for data and command access. You are advised to create **database accounts** and configure least-privilege permissions for the accounts. If any account permission does not meet the role requirements, update the account permission or **delete** the account. RDS for PostgreSQL has some **built-in accounts**, which are used to provide background O&M services for DB instances and cannot be used or deleted by users.

Enabling Data Backup

When you create an RDS DB instance, an automated backup policy is enabled by default with the retention period set to seven days. You can change the backup retention period as required. RDS for PostgreSQL DB instances support **automated backups** and **manual backups**. You can periodically back up your instance. If the instance fails or data is damaged, **restore it using backups** to ensure data reliability. For details, see [Data Backups](#).

Enabling Database Audit

By using the PostgreSQL Audit extension (pgAudit) with your RDS for PostgreSQL instance, you can capture detailed records that auditors usually need to meet compliance regulations. For example, you can use pgAudit to track changes made to specific databases and tables, as well as record users who make such changes and many other details. pgAudit is disabled by default. Enable it as required. For details, see [Using pgAudit](#).

Avoiding Binding an EIP to Your RDS for PostgreSQL Instance

Do not deploy your instance on the Internet or in a demilitarized zone (DMZ). Instead, deploy it on a Huawei Cloud private network and use routers or firewalls to control access to your instance. Do not bind an EIP to your instance to prohibit

unauthorized access and DDoS attacks from the Internet. If you have bound an EIP to your instance, you are advised to unbind it. If you do need an EIP, [configure security group rules](#) to restrict the source IP addresses that can access your instance.

Updating the Database Version to the Latest

PostgreSQL 9.5, 9.6, and 10 have reached end of life (EOL) and are no longer maintained by the community. [An EOS notice](#) has been released for RDS for PostgreSQL 9.5 and 9.6. Using an earlier version may pose security risks. Running the software of the latest version can protect the system from certain attacks. You can upgrade [the minor version](#) or [the major version](#) of your DB instance as required.

Configuring the Delay for Account Authentication Failures

By default, RDS for PostgreSQL instances have a built-in `auth_delay` extension. `auth_delay` causes the server to stop for a short period of time before an authentication failure message is returned, making it more difficult to crack the database password. To configure the delay for account authentication failures, change the value of the `auth_delay.milliseconds` parameter (which indicates the number of milliseconds to wait before reporting an authentication failure) by referring to [Modifying Parameters of an RDS for PostgreSQL Instance](#). The default value of this parameter is `0`.

4 RDS for SQL Server

4.1 Restoring Data from Backup Files to RDS for SQL Server DB Instances

This section describes how to use automated or manual backup files to restore a DB instance to the status when the backup was created.

Best Practices

- You can create a full backup file for your DB instance and use OBS and DRS to restore the backup file to an RDS for SQL Server DB instance.
- The restoration must be from an earlier database version to the same or a later version. The version of local backups must be earlier than or the same as the version of the destination DB instance to be restored.

NOTE

For example, if the local database version is Microsoft SQL Server 2012 Standard Edition, you can only restore the local backups to Standard or Enterprise Edition of Microsoft SQL Server 2014 or 2016. You cannot restore the local backups to any versions of Microsoft SQL Server 2008 or Web Editions of Microsoft SQL Server 2014 and 2016.

- For operation details on the RDS console, see [Restoring a DB Instance to a Point in Time](#) and [Restoring a DB Instance from a Backup](#).

4.2 Migrating Data from a Self-Managed SQL Server Database on an ECS to an RDS for SQL Server DB Instance

Scenarios

- You have created a Microsoft SQL Server database on an ECS.
- The self-managed SQL Server database version on the ECS cannot be later than the version of the RDS for SQL Server DB instance.

- You have installed the SQL Server Management Studio (SSMS).

Procedure

Step 1 Create an ECS.

NOTE

The ECS and the RDS DB instance must be in the same region and VPC.

Step 2 Install Microsoft SQL Server 2008, 2012, or 2014 on the ECS.

NOTE

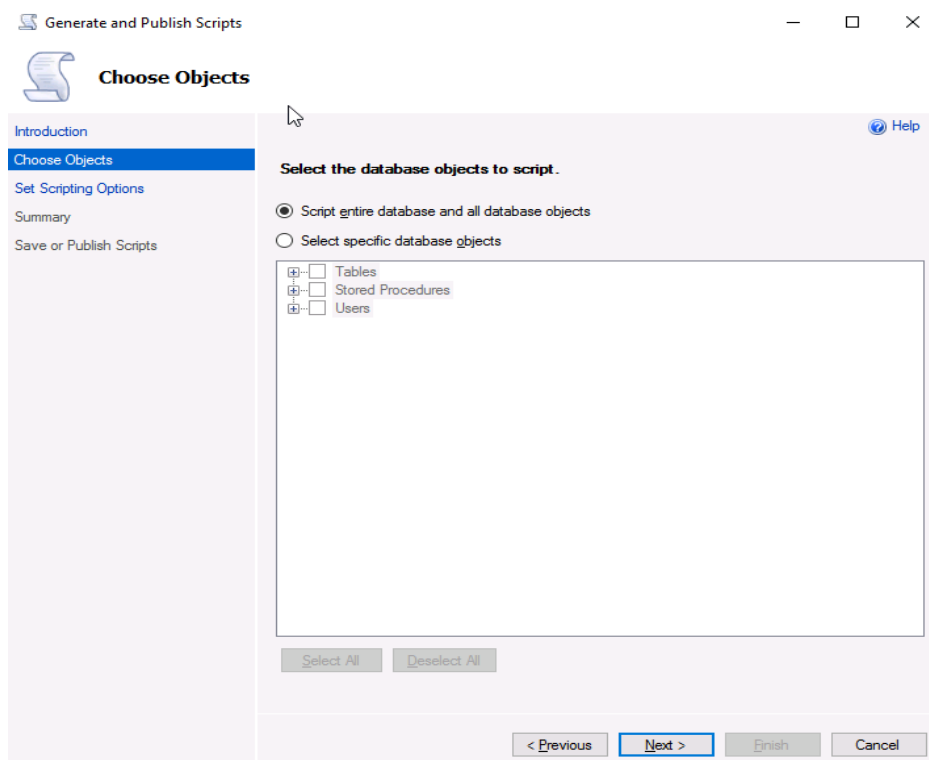
The Microsoft SQL Server installed on the ECS must be Standard or Enterprise Edition. It is recommended that the Microsoft SQL Server version be the same as the RDS DB instance version.

Step 3 Upload a local .bak file to the ECS and use Microsoft SQL Server to restore the local file to the RDS DB instance.

Step 4 Use the script generation tool provided by Microsoft SQL Server to generate a database structure script.

1. Right-click the database whose schema script needs to be generated and choose **Tasks > Generate Scripts**.
2. On the **Choose Objects** page, choose database objects to script, as shown in [Figure 4-1](#). Then, click **Next**.

Figure 4-1 Choosing objects

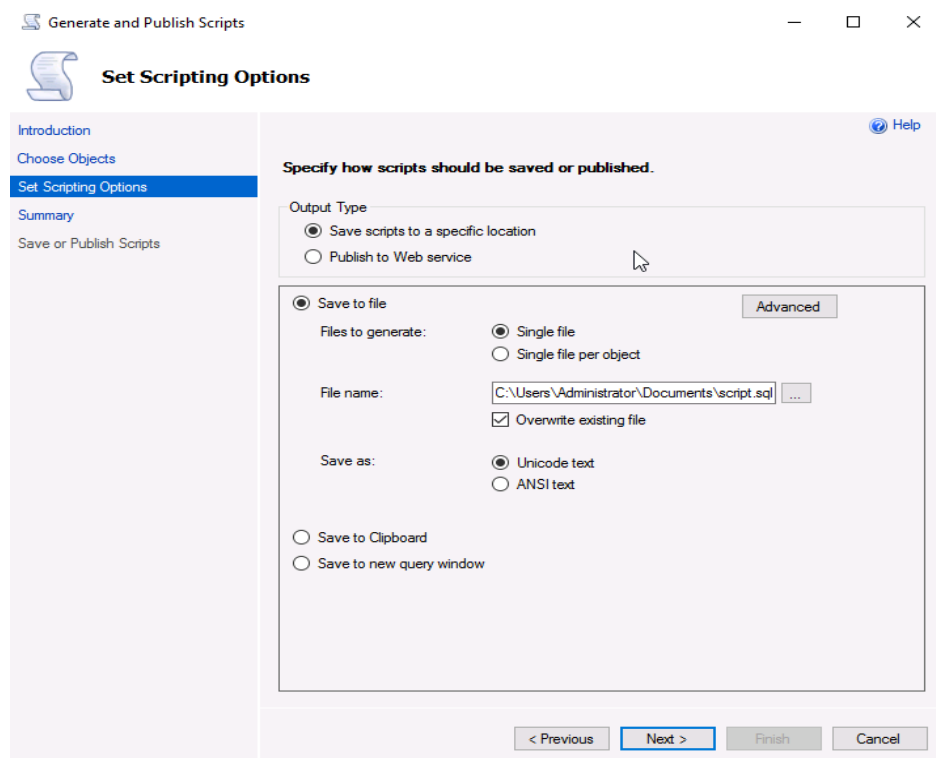


3. On the **Set Scripting Options** page, specify a directory for saving the script.

 **NOTE**

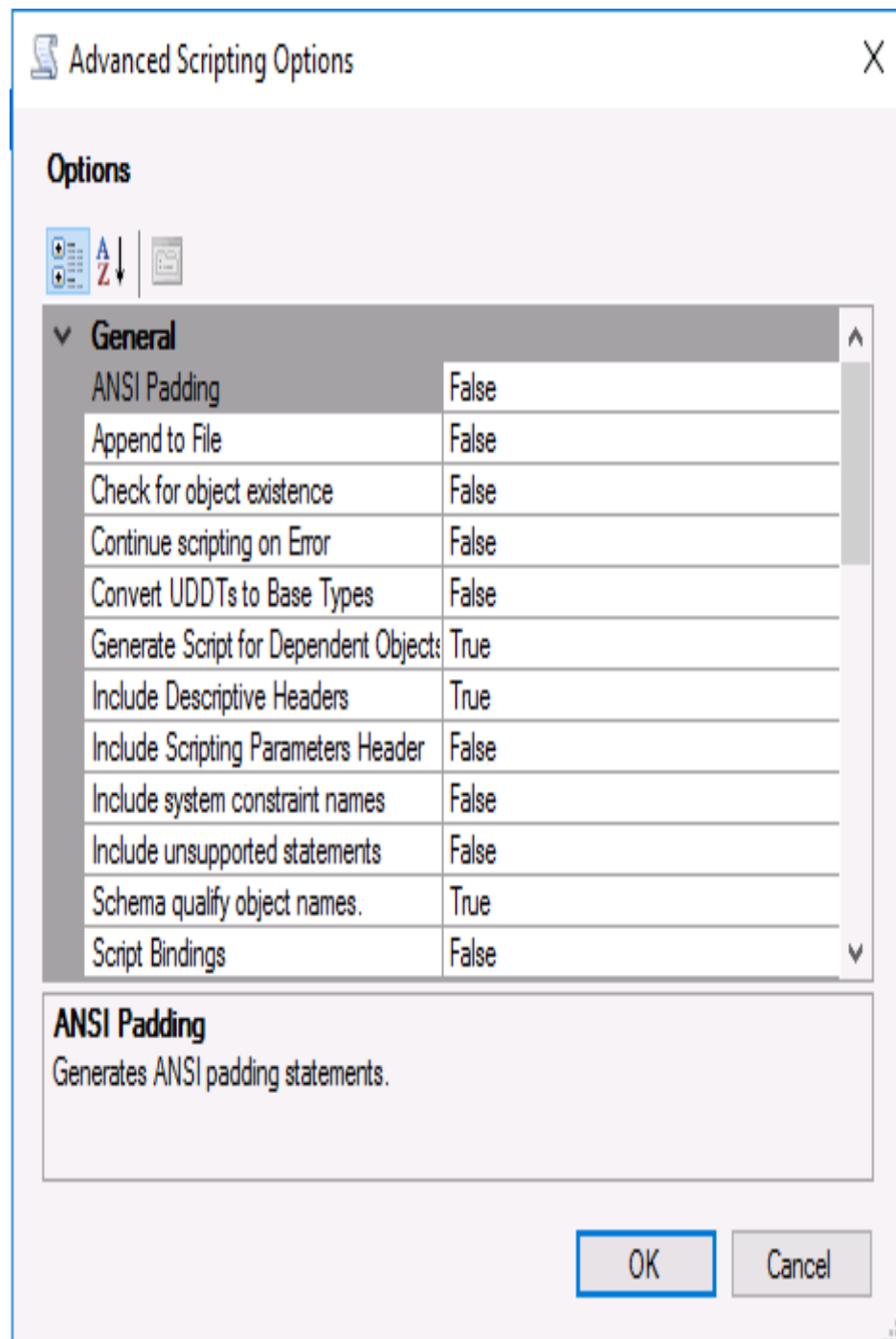
You are advised to save the script locally and generate an SQL script for execution.

Figure 4-2 Specifying a directory for saving the script



4. Click **Advanced**. In the displayed **Advanced Scripting Options** dialog box, specify scripting options for triggers, indexes, unique keys, the primary key, and server version. Then, click **OK**.

Figure 4-3 Specifying advanced scripting options



NOTE

Generate Script for Dependent Objects indicates the script data type option.

5. Click **Next** to generate the script.

Step 5 Use the SSMS client to connect to the RDS DB instance and open the generated SQL script.

 **NOTE**

You need to create an empty database, and then use the script to create structures in the database.

- Step 6** Use the import and export function provided by Microsoft SQL Server to migrate data.
1. Right-click the database where data is to be imported and choose **Tasks > Import Data**.
 2. Click **Next**.
 3. On the **Choose a Data Source** page, select a data source and click **Next**.
 4. On the **Choose a Destination** page, select a destination database and click **Next**.
 - **Destination:** Select **SQL Server Native Client** (depending on your destination database type).
 - **Server name:** Enter the IP address and port number of the destination DB instance.
 - **Authentication:** Select **Use SQL Server Authentication**. Then, set **User name** to **rdsuser**, and **Password** to the password of **rdsuser**.
 - **Database:** Select the destination database where data is to be imported.
 5. Select **Copy data from one or more tables or views** and click **Next**.
 6. On the **Select Source Tables and Views** page, select the tables and views that you want to copy. Then, click **Edit Mappings**. In the displayed dialog box, select **Enable identity insert** and edit mappings based on your requirements.
 7. Click **Next**.
 8. Select **Run immediately** and click **Next**.
 9. Click **Finish** to import data. You can view the progress. About 4,000 rows can be processed per second.



----End

4.3 Modifying Parameters of RDS for SQL Server Instances

You can modify parameters in custom parameter templates.

Each DB instance is assigned with a group of parameters when it is being created and modifications to these parameters do not affect other DB instances.

Procedure

- Step 1** [Log in to the management console](#).
- Step 2** Click  in the upper left corner and select a region.
- Step 3** Click  in the upper left corner of the page and choose **Databases > Relational Database Service**.

Step 4 On the **Instances** page, click the target DB instance.

Step 5 On the **Parameters** page, modify the parameters as required.

 **NOTE**

- You cannot modify parameters in a default parameter template.
 - Each Microsoft SQL Server version has a unique default parameter template.
 - To apply a default parameter template to the current DB instance, choose **Parameter Templates** page in the navigation pane on the left, locate the target template on the **Default Templates** page, and click **Apply** in the **Operation** column.
- You can modify parameters in a custom template.
 - To create a custom template, choose **Parameter Templates** page in the navigation pane on the left, click **Create Parameter Template** on the **Custom Templates** page, and configure required information in the displayed dialog box. Then, click **OK**.
 - After you save modifications to the parameters in the custom template, you can apply this parameter template to multiple DB instances running corresponding versions.

You can modify the parameters listed in [Table 4-1](#) to improve DB instance performance.

Table 4-1 Parameters

Parameter	Description	Application Scenario
max degree of parallelism	Specifies the maximum degree of parallelism option. When an RDS for SQL Server DB instance runs on a computer with more than one microprocessor or CPU, RDS for SQL Server detects the best degree of parallelism (the number of processors used to run a single statement) for each parallel plan execution. The default value is 0 .	<ul style="list-style-type: none"> • If the DB instance is used for querying results, set the parameter to 0. • If the DB instance is used for operations such as inserting, updating, and deleting, set the parameter to 1.
max server memory (MB)	Specifies the server memory option. It is used to reconfigure the amount of memory (in MB) in the buffer pool used by a Microsoft SQL Server DB instance.	<p>You are advised to retain the default value for this parameter.</p> <p>If you want to modify this parameter, the value of this parameter must be:</p> <ul style="list-style-type: none"> • No less than 2 GB. • Not greater than 95% of the maximum memory of the DB instance.

Parameter	Description	Application Scenario
user connections	Specifies the maximum number of simultaneous user connections allowed on Microsoft SQL Server. Default value: 1000	<ul style="list-style-type: none"> • If this parameter is set to 0, the number of connections to the DB instance is not limited. • Allowed values: Values excluding 1 to 10

- To save the modifications, click **Save**.
- To cancel the modifications, click **Cancel**.
- To preview the modifications, click **Preview**.

After the parameter values are modified, you can click **Change History** to view the modification details.

----End

4.4 Supporting DMVs

RDS for SQL Server supports dynamic management views (DMVs), which enables users to quickly find 10 SQL statements with the highest performance consumption.

Scenarios

- A performance bottleneck occurs and the database execution efficiency becomes low.
- The monitoring result shows that the CPU and I/O are high in some time segments.

Procedure

- Step 1** Use the **rdsuser** account to connect to the target DB instance through a client and run the following statements on the management plane:

```
declare @DatabaseName nvarchar(100)
set @DatabaseName = 'Wisdom_TT_ODS'

select top 10
DB_NAME(st.dbid) as DBName, OBJECT_NAME(st.objectid,st.dbid) as ObjectName,
substring(st.text,(qs.statement_start_offset/2)+1,((case qs.statement_end_offset when -1 then
datalength(st.text) else qs.statement_end_offset end - qs.statement_start_offset)/2) + 1) as
Statement,
st.text as Query,
qp.query_plan,
plan_generation_num,
creation_time,
last_execution_time,
execution_count,
total_worker_time,
```

```

min_worker_time,
max_worker_time,
total_logical_reads,
min_logical_reads,
max_logical_reads,
total_elapsed_time,
min_elapsed_time,
max_elapsed_time,
total_rows,
min_rows,
max_rows,
total_worker_time/execution_count as avg_worker_time,           --Average CPU
duration
total_logical_reads/execution_count as avg_logical_reads,       --Average logical reads
total_elapsed_time/execution_count as avg_elapsed_time,         --Average total
duration
total_rows/execution_count as avg_rows,                           --Average data processing
rows
sql_handle,
plan_handle,
query_hash,
query_plan_hash
from sys.dm_exec_query_stats qs
cross apply sys.dm_exec_sql_text(plan_handle) st
cross apply sys.dm_exec_query_plan(plan_handle) qp
where st.dbid=DB_ID(@DatabaseName)
and text not like '%sys.%'and text not like '%[[sys]%'
order by avg_worker_time desc

```

Step 2 You can view the SQL execution records and resource consumption details of the corresponding database in the query result.

----End


4.5 Using the Import and Export Function to Migrate Data from a Local Database to an RDS for SQL Server DB Instance


Scenarios

- You have created a local Microsoft SQL Server database.
- The local database version cannot be later than the version of the destination RDS for SQL Server DB instance.
- You want to migrate only tables instead of the whole database.

Procedure

Step 1 [Log in to the management console.](#)

Step 2 Click  in the upper left corner and select a region.

Step 3 Click  in the upper left corner of the page and choose **Databases > Relational Database Service**.

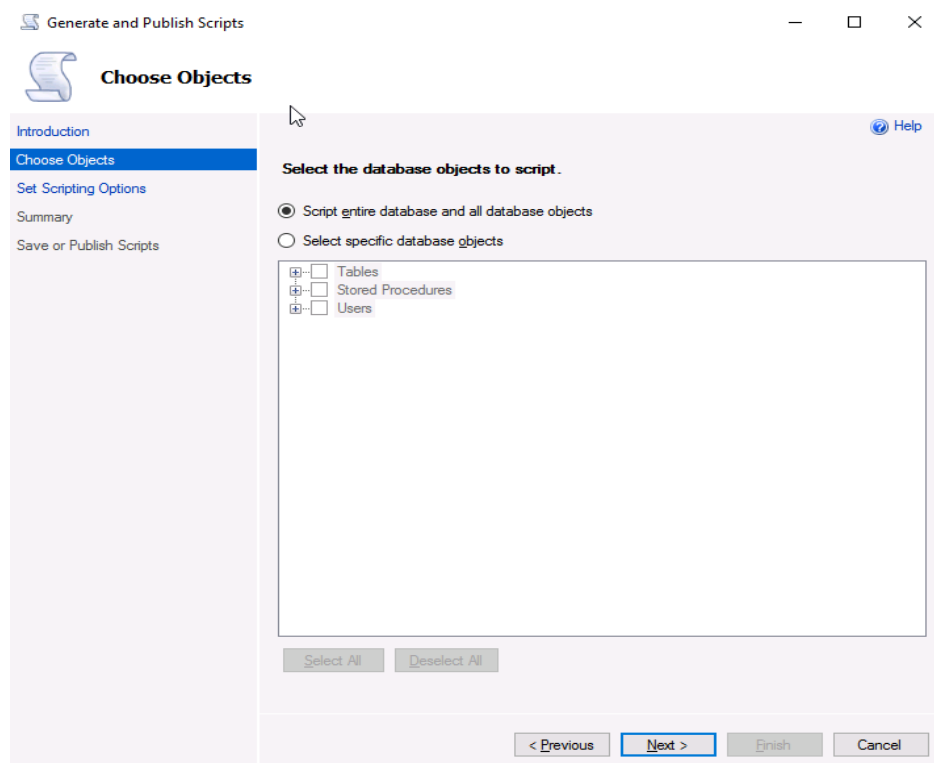
- Step 4** On the **Instances** page, click the instance name.
- Step 5** In the navigation pane on the left, choose **Connectivity & Security**.
- Step 6** In the **Connection Information** area, click **Bind** next to the **EIP** field.
- Step 7** In the displayed dialog box, select an EIP and click **Yes**.
- Step 8** Install the SSMS client locally and use the EIP to connect to the RDS DB instance.

 **NOTE**

Click [here](#) to download the SSMS client.

- Step 9** Use the script generation tool provided by Microsoft SQL Server to generate a database structure script.
 1. Right-click the database whose schema script needs to be generated and choose **Tasks > Generate Scripts**.
 2. On the **Choose Objects** page, choose database objects to script, as shown in **Figure 4-4**. Then, click **Next**.

Figure 4-4 Choosing objects

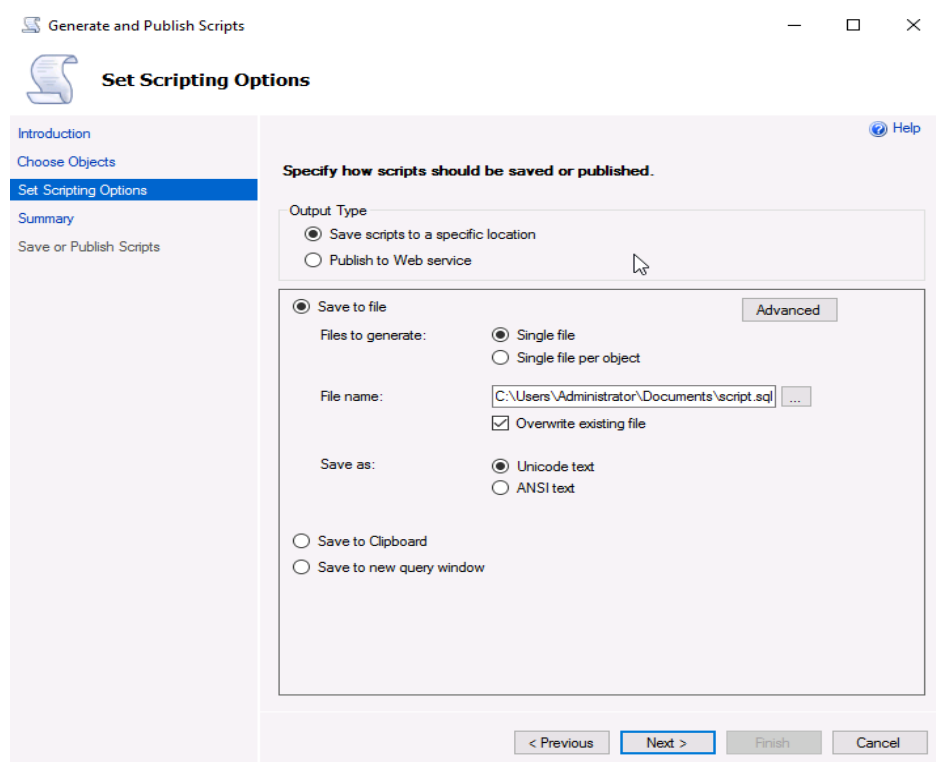


- 3. On the **Set Scripting Options** page, specify a directory for saving the script.

 **NOTE**

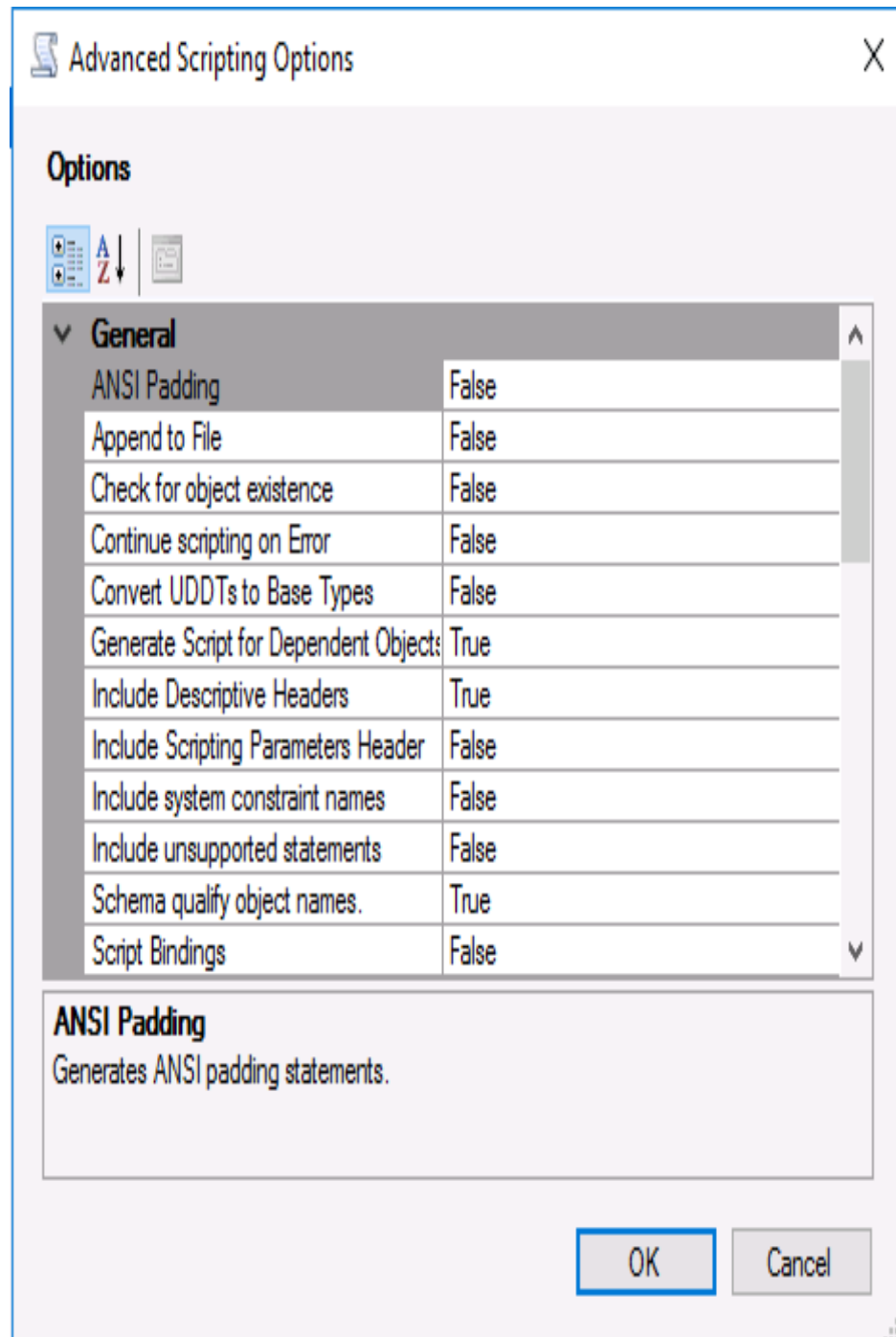
You are advised to save the script locally and generate an SQL script for execution.

Figure 4-5 Specifying a directory for saving the script



4. Click **Advanced**. In the displayed **Advanced Scripting Options** dialog box, specify scripting options for triggers, indexes, unique keys, the primary key, and server version. Then, click **OK**.

Figure 4-6 Specifying advanced scripting options



NOTE

Generate Script for Dependent Objects indicates the script data type option.

5. Click **Next** to generate the script.

Step 10 Use the SSMS client to connect to the RDS DB instance and open the generated SQL script.

 NOTE

You need to create an empty database, and then use the script to create structures in the database.

- Step 11** Use the import and export function provided by Microsoft SQL Server to migrate data.
1. Right-click the database where data is to be imported and choose **Tasks > Import Data**.
 2. Click **Next**.
 3. On the **Choose a Data Source** page, select a data source and click **Next**.
 4. On the **Choose a Destination** page, select a destination database and click **Next**.
 - **Destination:** Select **SQL Server Native Client** (depending on your destination database type).
 - **Server name:** Enter the IP address and port number of the destination DB instance.
 - **Authentication:** Select **Use SQL Server Authentication**. Then, set **User name** to **rdsuser**, and **Password** to the password of **rdsuser**.
 - **Database:** Select the destination database where data is to be imported.
 5. Select **Copy data from one or more tables or views** and click **Next**.
 6. On the **Select Source Tables and Views** page, select the tables and views that you want to copy. Then, click **Edit Mappings**. In the displayed dialog box, select **Enable identity insert** and edit mappings based on your requirements.
 7. Click **Next**.
 8. Select **Run immediately** and click **Next**.
 9. Click **Finish** to import data. You can view the progress. About 4,000 rows can be processed per second.

----End

4.6 Creating a Subaccount of rdsuser

Scenarios

This section describes how to create a subaccount and grant permissions to the subaccount. [Permissions of rdsuser](#) lists the permissions supported by **rdsuser**.

Prerequisites

You have created a database. For details, see [Creating a Database](#).

Procedure

Step 1 Log in to the instance through DAS.

1. [Log in to the management console](#).



2. Click  in the upper left corner and select a region.
3. Click  in the upper left corner of the page and choose **Databases > Relational Database Service**.
4. Locate the target instance and click **Log In** in the **Operation** column.
5. On the displayed page, configure required parameters.

Table 4-2 Instance login

Parameter	Description
Login Username	Enter rdsuser .
Password	Enter the password of rdsuser . NOTE You can select Remember Password so that you can directly log in to the instance next time.
Collect Metadata Periodically	Enable this function as required. If this function is enabled: <ul style="list-style-type: none"> - DAS can store structure definition data such as database names, table names, and field names in instances, but does not store data in tables. - Metadata is collected in the early morning every day.
Show Executed SQL Statements	Enable this function as required. This function allows you to view executed SQL statements. You can re-execute an SQL statement without having to enter it again.

6. Click **Log In**.

Step 2 Create a subaccount.

1. On the main menu of the DAS console, choose **Account Management > Login Name**.
2. On the displayed page, click **Create Login Name**.
3. On the **Create Login Name** page, configure login information.

Table 4-3 Login information

Parameter	Description
Login Name	Enter a new login name.
Authentication Type	The value is fixed to Microsoft SQL Server Authentication .

Parameter	Description
Password	The password for the new login username must meet the following requirements: <ul style="list-style-type: none"> - It must contain at least three of the following: uppercase letters, lowercase letters, digits, and special characters ~!@#%^*_+=+?% - It must be 8 to 128 characters long. - It cannot contain the login username. - It cannot be a weak password.
Confirm Password	Enter the password again. NOTE For security purposes, select Enforce Password Policy .
Default Database	From the drop-down list, select a database the new login user will log in by default.
Default Language	Select a language for the new login user.

4. Click **Save**.
5. Click **Back to Login Name List**.
6. In the login name list, view the new login name.

Step 3 Grant permissions to the new login user.

 **NOTE**

- [Table 4-4](#) describes how to add a single permission. To add multiple permissions to the new login user at the same time, for example, to grant both read and write permissions, select both **db_datareader** and **db_datawriter** on the **Edit Database Role** page.
- For details about the permissions supported by **rdsuser**, see [Permissions of rdsuser](#).

Table 4-4 Permissions that can be granted to a subaccount

Permission	Procedure
Database operation permission	<ol style="list-style-type: none"> 1. Locate the new login name and click Edit in the Operation column. 2. On the displayed page, click the User Mapping tab. 3. In the Users mapped to this login list, click Edit in the row where both the user database and the new login username exist. 4. On the Edit Database Role page, select db_owner and click OK. 5. Click Save.

Permission	Procedure
Server role permission	<ol style="list-style-type: none"> 1. Locate the new login name and click Edit in the Operation column. 2. On the displayed page, click the Server Roles tab. 3. In the Server Roles list, select the desired server role. 4. Click Save.
Securable object permission	<ol style="list-style-type: none"> 1. Locate the new login name and click Edit in the Operation column. 2. On the displayed page, click the Securables tab. 3. In the securables list, select the desired server permission. 4. Click Save.
Read-only permission	<ol style="list-style-type: none"> 1. Locate the new login name and click Edit in the Operation column. 2. On the displayed page, click the User Mapping tab. 3. In the Users mapped to this login list, click Edit in the row where both the user database and the new login username exist. 4. On the Edit Database Role page, select db_datareader and click OK. 5. Click Save.
Write permission	<ol style="list-style-type: none"> 1. Locate the new login name and click Edit in the Operation column. 2. On the displayed page, click the User Mapping tab. 3. In the Users mapped to this login list, click Edit in the row where both the user database and the new login username exist. 4. On the Edit Database Role page, select db_datawriter and click OK. 5. Click Save.

----End

Permissions of rdsuser

Table 4-5 Permissions of rdsuser

Name	Category	Permission
DB instance permissions	DB instance role permissions	[processadmin]
		[setupadmin]
	DB instance object permissions	ALTER ANY CONNECTION
		ALTER ANY LOGIN
		ALTER ANY SERVER ROLE
		ALTER SERVER STATE
		ALTER TRACE
		CONNECT ANY DATABASE
		CONTROL SERVER
		CONNECT SQL
		CREATE ANY DATABASE
		SELECT ALL USER SECURABLES
		VIEW ANY DEFINITION
		VIEW ANY DATABASE
		VIEW SERVER STATE
	Database permissions	master: Public
		MsdB: Public SQLAgentUserRole
		Model: Public
		Rdsadmin: Public
		OtherDB: Db_Owner

4.7 Creating tempdb Files

Scenarios

The tempdb system database is a global resource that is available to all users connected to an instance of SQL Server or SQL Database. It is a temporary database that cannot store data permanently. It is used to process intermediate data for various requests in the instance. Physical properties of tempdb in SQL

Server are classified into the primary data files (.mdf), secondary data files (.ndf), and log files (.ldf). **tempdb** is re-created every time SQL Server is started.

There may be some issues or even service interruption if applications frequently create and drop tempdb files, especially in high-concurrency scenarios.

Microsoft recommends that the tempdb files be divided into multiple files. Generally, the number of files depends on the number of vCPUs (logical). If the number of vCPUs is greater than eight, use eight data files and then if contention continues, increase the number of data files by multiples of 4 until the contention is reduced to acceptable levels or make changes to the workload/code.

For more information, see [tempdb Database](#) in the Microsoft official website.

Constraints

- By default, each RDS for SQL Server instance running SQL Server 2008, 2012, or 2014 Edition has one tempdb file, each instance running SQL Server 2016 Edition has four tempdb files, and each instance running SQL Server 2017 Edition has eight tempdb files.
- Each RDS for SQL Server instance has only one log file no matter which SQL Server Edition they run.

Application Scenario

You need to determine the number of tempdb files to be created based on the instance specifications and scenarios. The following uses an example to show how to create 8 tempdb files for a SQL Server 2014 Enterprise Edition instance with 32 vCPUs.

Prerequisites

- Visit the [Microsoft](#) website and obtain the installation package of SQL Server Management Studio. Double-click the installation package and complete the installation as instructed.
- You have created an instance with 32 vCPUs running Microsoft SQL Server 2014 Enterprise Edition. For details, see [Buy a DB Instance](#)

Procedure

Step 1 Start SQL Server Management Studio.

Step 2 Choose **Connect > Database Engine**. In the displayed dialog box, enter login information.

Figure 4-7 Connecting to the server

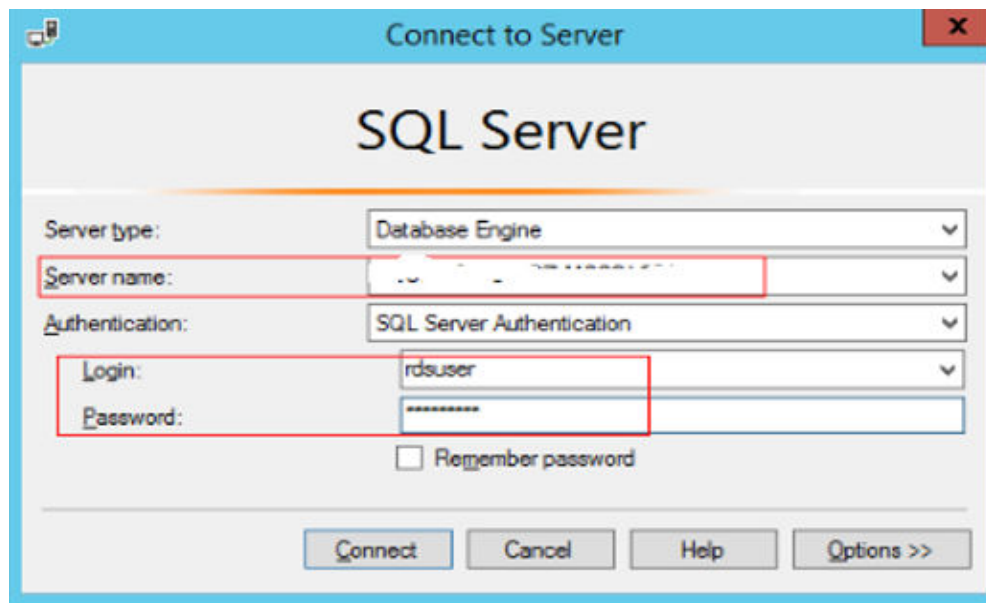


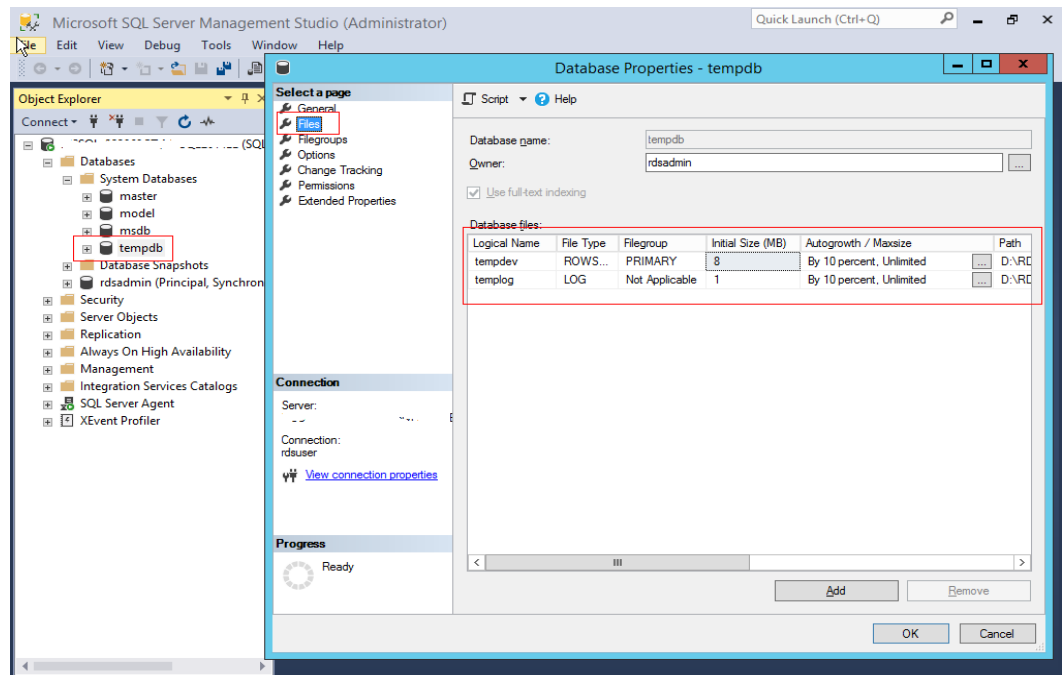
Table 4-6 Parameter description

Parameter	Description
Server name	Indicates the IP address and database port of the DB instance. Use a comma (,) to separate them. For example: x.x.x.x,8080. <ul style="list-style-type: none"> • The IP address is the EIP that has been bound to the DB instance. • The database port is that displayed on the Connectivity & Security page.
Authentication	Indicates the authentication mode. Select SQL Server Authentication .
Login	Indicates the database account used for accessing the instance. The default administrator account is rdsuser .
Password	Indicates the password of the database account.

Step 3 View the tempdb information.

- Choose **Databases > System Databases > tempdb**. Right-click **tempdb** and choose **Database Properties**. In the displayed dialog box, view the current tempdb information.

Figure 4-8 Viewing current tempdb information



- You can also run the following SQL statements to query the tempdb information:

```

SELECT name AS FileName,
size*1.0/128 AS FileSizeInMB,
CASE max_size
WHEN 0 THEN 'Autogrowth is off.'
WHEN -1 THEN 'Autogrowth is on.'
ELSE 'Log file grows to a maximum size of 2 TB.'
END,
growth AS 'GrowthValue',
'GrowthIncrement' =
CASE
WHEN growth = 0 THEN 'Size is fixed.'
WHEN growth > 0 AND is_percent_growth = 0
THEN 'Growth value is in 8-KB pages.'
ELSE 'Growth value is a percentage.'
END
FROM tempdb.sys.database_files;
GO

```

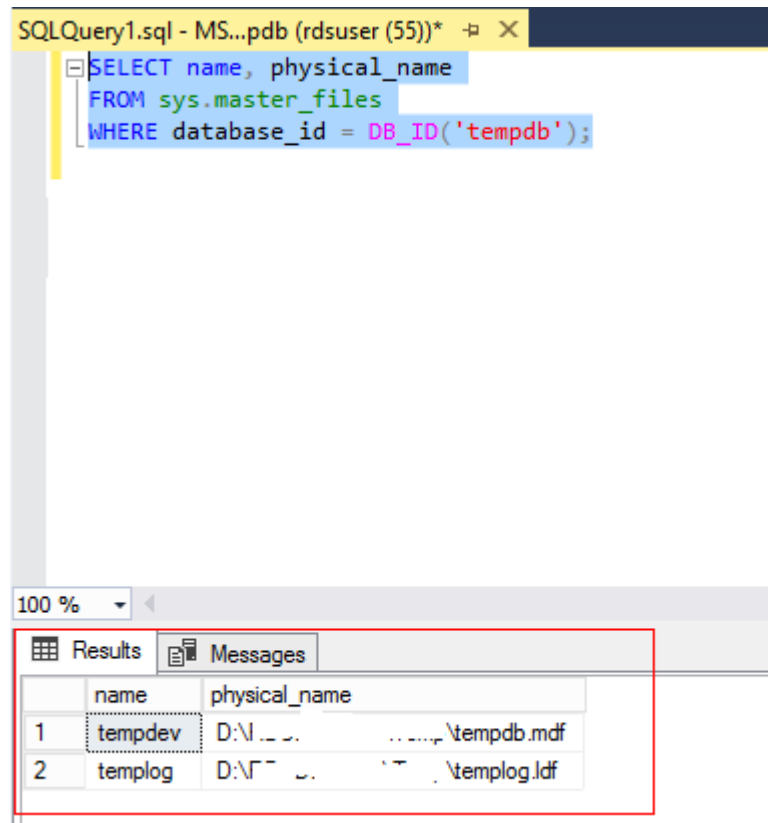
- Step 4** Run the following statements to query the tempdb file name of the current DB instance:

```

SELECT name, physical_name
FROM sys.master_files
WHERE database_id = DB_ID('tempdb');

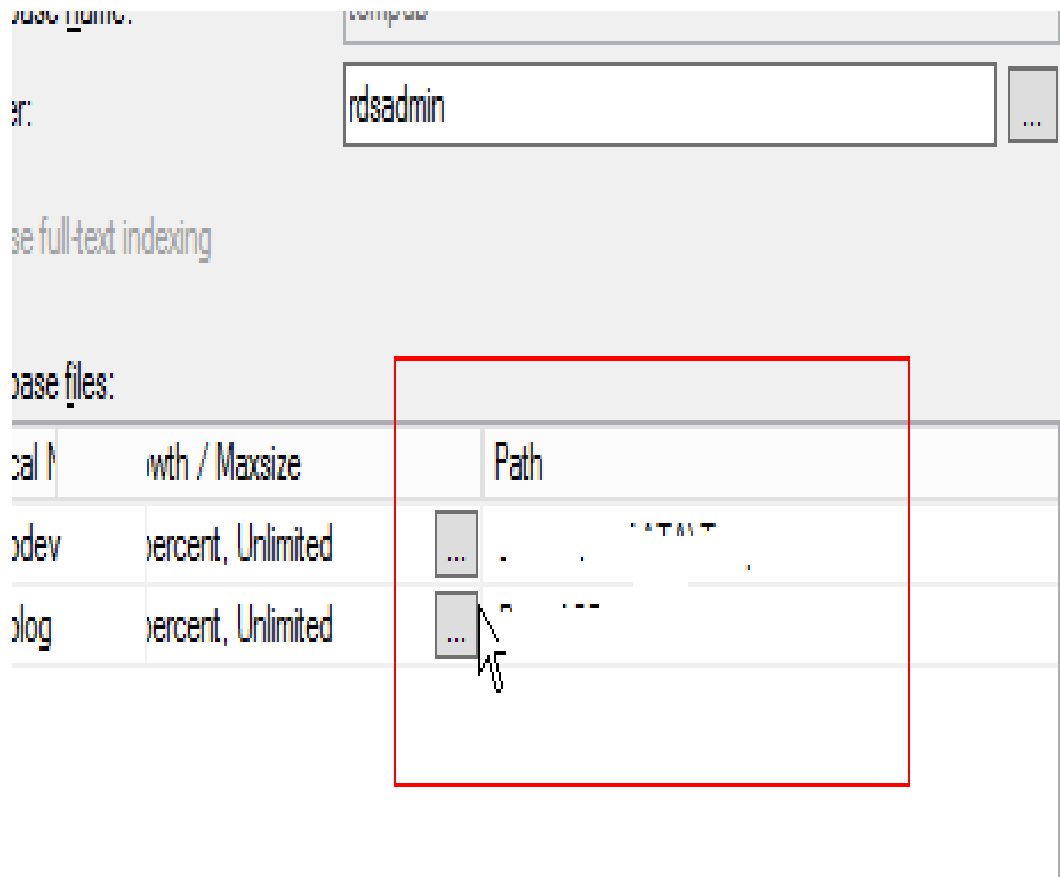
```

Figure 4-9 Viewing tempdb file names



Step 5 On the **Files** tab in **Step 3**, view the paths of tempdb files.

Figure 4-10 Viewing tempdb paths



Step 6 Run the following statements to migrate the tempdb files to **D:\RDSDBDATA\DATA** and specify the initial size and growth as required.

USE master;

GO

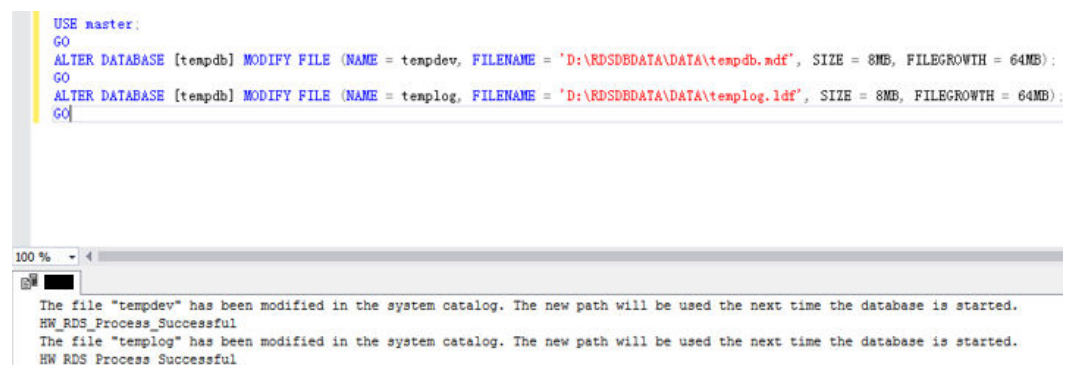
ALTER DATABASE [tempdb] MODIFY FILE (NAME = tempdev, FILENAME = 'D:\RDSDBDATA\DATA\tempdev.mdf', SIZE = 8MB, FILEGROWTH = 64MB);

GO

ALTER DATABASE [tempdb] MODIFY FILE (NAME = templog, FILENAME = 'D:\RDSDBDATA\DATA\templog.ldf', SIZE = 8MB, FILEGROWTH = 64MB);

GO

Figure 4-11 Moving tempdb files



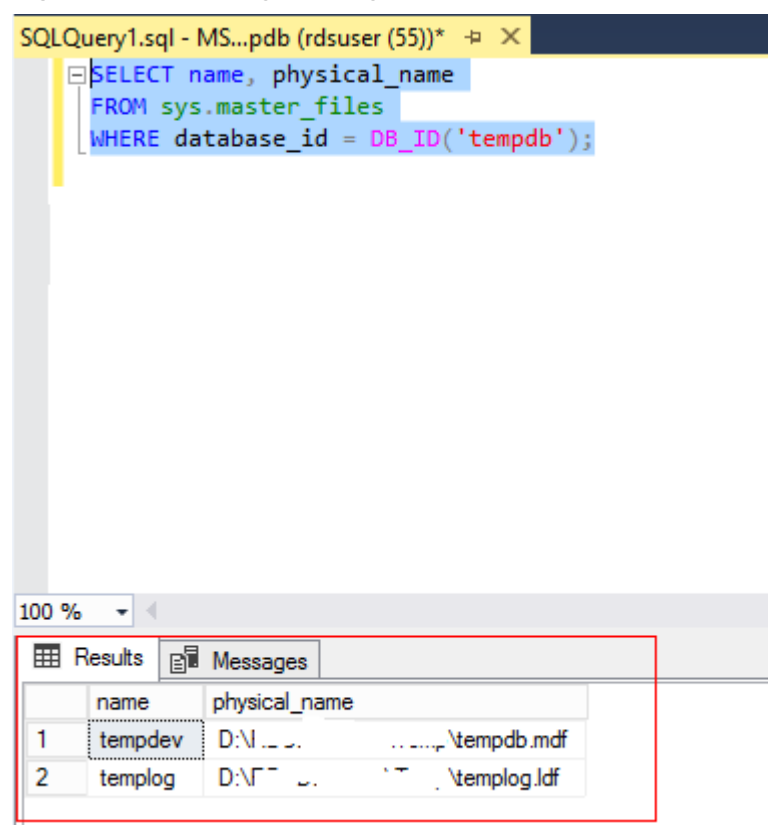
Step 7 On the **Instances** page of the RDS console, locate the target DB instance and choose **More > Reboot** in the **Operation** column to reboot the DB instance.

You can also click the target DB instance. On the displayed page, click **Reboot** in the upper right corner of the page.

Step 8 Run the following SQL statements to check whether the tempdb files are successfully migrated:

```
SELECT name, physical_name
FROM sys.master_files
WHERE database_id = DB_ID('tempdb');
```

Figure 4-12 Viewing the migration results



Step 9 Configure the file name, initial size, and growth as required. Add tempdb files using either of the following methods:

- Adding tempdb files through SQL statements

Based on the number of vCPUs and tempdb files to be added, set the initial size to 8 MB and file growth to 64 MB.

```
USE [master]
```

```
GO
```

```
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'temp2', FILENAME =  
N'D:\RDSDBDATA\DATA\tempdb2.ndf', SIZE = 8MB, FILEGROWTH = 64MB)
```

```
GO
```

```
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'temp3', FILENAME =  
N'D:\RDSDBDATA\DATA\tempdb3.ndf', SIZE = 8MB, FILEGROWTH = 64MB)
```

```
GO
```

```
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'temp4', FILENAME =  
N'D:\RDSDBDATA\DATA\tempdb4.ndf', SIZE = 8MB, FILEGROWTH = 64MB)
```

```
GO
```

```
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'temp5', FILENAME =  
N'D:\RDSDBDATA\DATA\tempdb5.ndf', SIZE = 8MB, FILEGROWTH = 64MB)
```

```
GO
```

```
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'temp6', FILENAME =  
N'D:\RDSDBDATA\DATA\tempdb6.ndf', SIZE = 8MB, FILEGROWTH = 64MB)
```

```
GO
```

```
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'temp7', FILENAME =  
N'D:\RDSDBDATA\DATA\tempdb7.ndf', SIZE = 8MB, FILEGROWTH = 64MB)
```

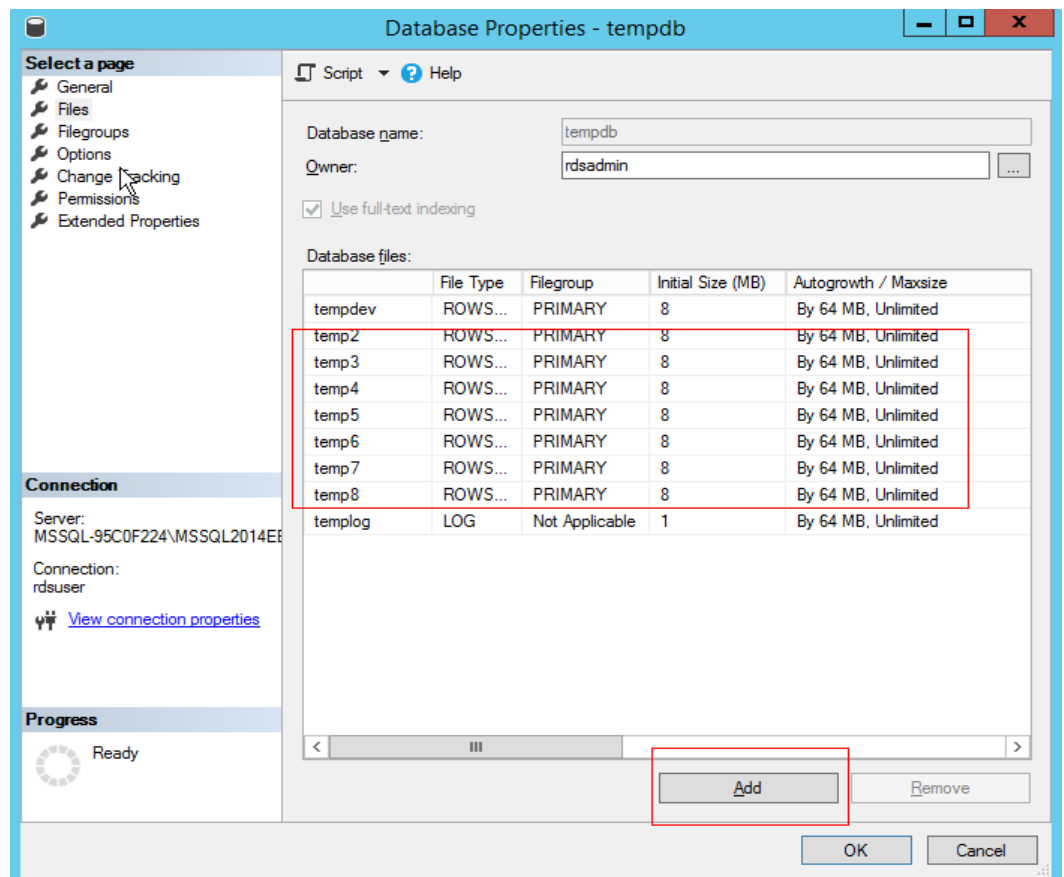
```
GO
```

```
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'temp8', FILENAME =  
N'D:\RDSDBDATA\DATA\tempdb8.ndf', SIZE = 8MB, FILEGROWTH = 64MB)
```

```
GO
```

- Adding tempdb files through SQL Server Management Studio On the **Files** tab in [Step 3](#), click **Add**.

Figure 4-13 Adding tempdb files through the client



Step 10 After the configuration is complete, reboot the DB instance again by referring to [Step 7](#).

Step 11 Repeat [Step 8](#) to check whether the tempdb files are successfully added.

Figure 4-14 Viewing the added tempdb files

	name	physical_name
1	tempdev	D:\... \DATA\tempdb.mdf
2	templog	D:\... \DATA\templog.ldf
3	temp2	D:\... \DATA\tempdb2.ndf
4	temp3	D:\... \DATA\tempdb3.ndf
5	temp4	D:\... \DATA\tempdb4.ndf
6	temp5	D:\... \DATA\tempdb5.ndf
7	temp6	D:\... \DATA\tempdb6.ndf
8	temp7	D:\... \DATA\tempdb7.ndf
9	temp8	D:\... \DATA\tempdb8.ndf

----End

4.8 Microsoft SQL Server Publication and Subscription

The publication and subscription function provided by Microsoft SQL Server uses the replication technology for data synchronization. This function makes it possible to split data read and write operations and synchronize offline and online data.

This section describes how to use SQL Server Management Studio (SSMS) to configure publication and subscription. You can create publications and subscriptions for RDS for SQL Server instances on the console. For details, see [Creating a Publication](#).

Preparations

Environments

1. Local environment: a local database running Microsoft SQL Server 2014 Standard Edition in Windows
2. Online environment:
 - One single DB instance with 2 vCPUs and 16 GB of memory running Microsoft SQL Server 2014 Standard Edition, bound with an EIP
 - One primary/standby DB instance with 4 vCPUs and 8 GB of memory running Microsoft SQL Server 2014 Standard Edition, bound with an EIP

Environment Setup

- Publisher: maintains source data and distributes specific data to the distributor. In this example, use the local server as the publisher.

- a. Use SSMS to log in to the local database as user **sa**. Right-click the **Replication** folder and then click **Configure Distribution**. You can use your local server as the distributor or use another server as the distributor. Click **Next**.

NOTICE

- **sa** is the administrator account.
 - The login account must have the **sysadmin** permission. Otherwise, publication and subscription cannot be configured.
-

- b. Specify a root snapshot folder and click **Next**.

 **NOTE**

Related agent permissions must be configured for the publication so that the agent account has the permissions to read from and write to the folder.

- c. Specify the names and directories of the distribution database and log files, and click **Next**.
 - d. Specify the distributor for the publisher and then click **Next**.
 - e. Click **Finish** to complete the configuration.
- Configuring the agent account control file
 - a. You need to add the agent account to the control property of the snapshot folder according to the folder directory. Otherwise, an error message will be displayed, indicating that the access is denied.
 - b. Open the local SQL Server Configuration Manager, right-click the corresponding agent, choose **Properties** from the shortcut menu, and copy the account name.
 - c. Return to the directory of the snapshot folder. Right-click the folder and choose **Properties** from the shortcut menu. In the displayed dialog box, choose **Security** > **Edit** > **Add**. Select the local path and agent account name, click **OK**, and select all permissions.
 - Distributor: distributes data to specific subscribers. In this example, the distributor and publisher share the same server. Therefore, no extra configuration is required. For more information, see [Configure Distribution](#) at the official website.
 - Subscriber: receives data from the distributor. Subscription includes push subscription and pull subscription.
 - Push subscription: The publisher propagates changes to a subscriber without a request from the subscriber. Changes can be pushed to subscribers continuously or on a frequently recurring schedule.
 - Pull subscription: The subscriber requests changes made at the publisher. The data is usually synchronized on demand or on a schedule rather than continuously. RDS for SQL Server instances do not support pull subscription. In this example, only push subscription can be configured.

Before the subscription, ensure that the RDS for SQL Server instance can be accessed from the local server.

Before configuring the local subscription, you need to configure the RDS instance information on the local server.

- a. Configure an alias name for the subscriber on the local server. The subscription service does not support IP address-based access. Therefore, you need to map the EIP of the RDS DB instance to an alias name. To obtain the alias name, log in to the RDS DB instance and run the following SQL statement:

```
select @@SERVERNAME
```
- b. After obtaining the alias name, open the local SQL Server Configuration Manager, select the native clients, right-click **Aliases**, and choose **New Aliases** from the shortcut menu.
- c. Enter related information and click **OK**.

Table 4-7 Parameter description

Parameter	Description
Alias Name	Alias name configured in a
Port No	Port number of the RDS instance
Server	EIP bound to the RDS instance

- d. In **C:\Windows\System32\drivers\etc**, open the host file and add a mapping:

```
Server_address MSSQL-177FFD84\MSSQL2014STD
```

Creating a Publication

Step 1 Create a publication.

Expand the **Replication** folder, and then right-click the **Local Publications** folder. Click **New Publication**.

Step 2 Select **Transactional publication**.

Step 3 Select a table as the publication object.

Step 4 Add the object to be filtered for personalized publication.

Step 5 Create a snapshot to replicate the current state of the table. You can also set up a snapshot agent to execute the plan.

Step 6 Configure the agent security. You need to set the login account to the local **sa** account.

Step 7 Configure the publication name and click **Finish**.

Step 8 Check whether the publication is created by using the replication monitor.

----End

Creating a Subscription

Step 1 Right-click the publication for which you want to create one or more subscriptions, and then select **New Subscriptions**.

- Step 2** Configure the required parameters and click **Next**.
 - Step 3** Select push subscription and click **Next**.
 - Step 4** Click **Add Subscriber**. Both the SQL Server engine and non-SQL Server engine can be used as subscribers. In this example, use the RDS for SQL Server instance as the subscriber.
 - Step 5** Select a database as the subscription object.
 - Step 6** Configure the connection to the subscriber.
 - Step 7** Use a database account that is valid for a long time to ensure the subscription validity. You can use the account for logging in to the RDS for SQL Server instance. Then, click **OK**.
 - Step 8** Check whether the subscription is created successfully.
 - Step 9** Move the cursor to the publication to view the subscription information.
- End

4.9 Installing a C# CLR Assembly in RDS for SQL Server

Microsoft SQL Server provides assemblies to make database operations simple and convenient.

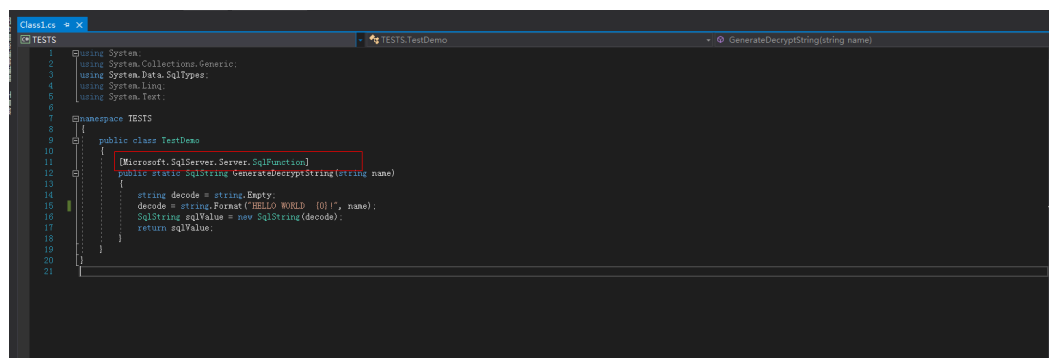
NOTE

When you restore data to a new or an existing DB instance, the **clr enabled** parameter is disabled by default. To use the CLR integration function, you need to enable **clr enabled** first. For details about how to enable the CLR integration function, see [Enabling CLR Integration](#).

Procedure

- Step 1** Create a C# function to compile an SQL Server DLL.

Figure 4-15 C# function code



```

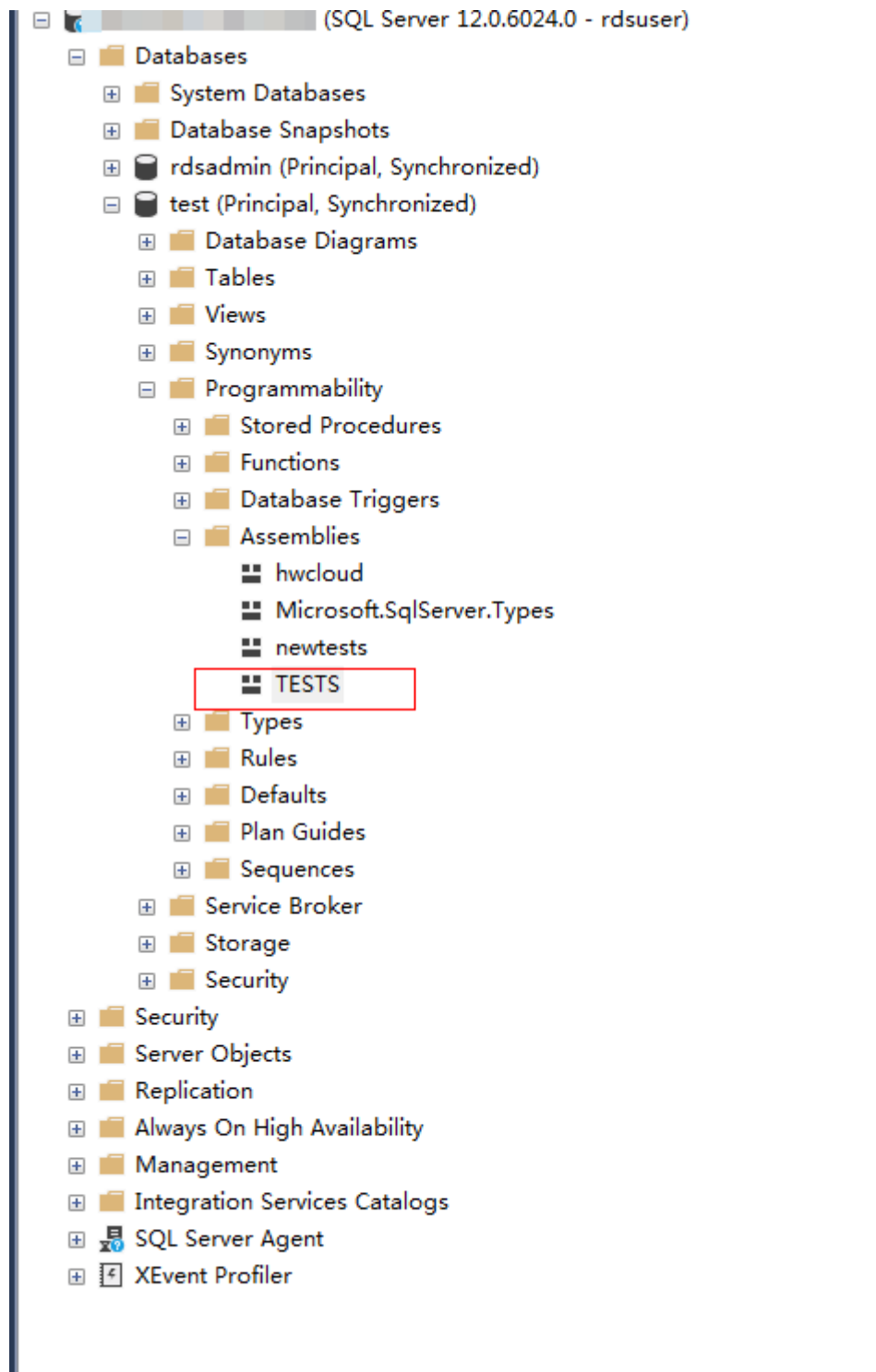
1 using System;
2 using System.Collections.Generic;
3 using System.Data.SqlTypes;
4 using System.Linq;
5 using System.Text;
6
7 namespace TESTS
8 {
9     public class TestDemo
10    {
11        [Microsoft.SqlServer.Server.SqlFunction]
12        public static SqlString GenerateDecryptString(string name)
13        {
14            string decode = string.Empty;
15            decode = string.Format("HELLO WORLD {0}!", name);
16            SqlString sqlValue = new SqlString(decode);
17            return sqlValue;
18        }
19    }
20 }

```

NOTICE

For more information about user-defined functions, see [CLR User-Defined Functions](#).

Figure 4-20 TESTS assembly



----End

4.10 Creating a Linked Server for an RDS for SQL Server DB Instance

Create a linked server for RDS for SQL Server DB instance named 2 to access another RDS for SQL Server DB instance named 1.

- Step 1** Enable distributed transactions of the two DB instances by referring to [Distributed Transactions](#) and add the peer-end host information to each other. For on-premises servers or ECSs, see [Resolving Names on Remote Servers \(ECSs\)](#).

 **NOTE**

RDS for SQL Server instance 2 and RDS for SQL Server instance 1 are in the same VPC. If the ECS and RDS instances are not in the same VPC or you are creating an on-premises server for an RDS instance, bind an EIP to the RDS instance. For details, see [Binding and Unbinding an EIP](#).

- Step 2** In DB instance 1, create database dbtest1 as user **rdsuser**.

- Step 3** In DB instance 2, run the following SQL statements to create a linked server as user **rdsuser**:

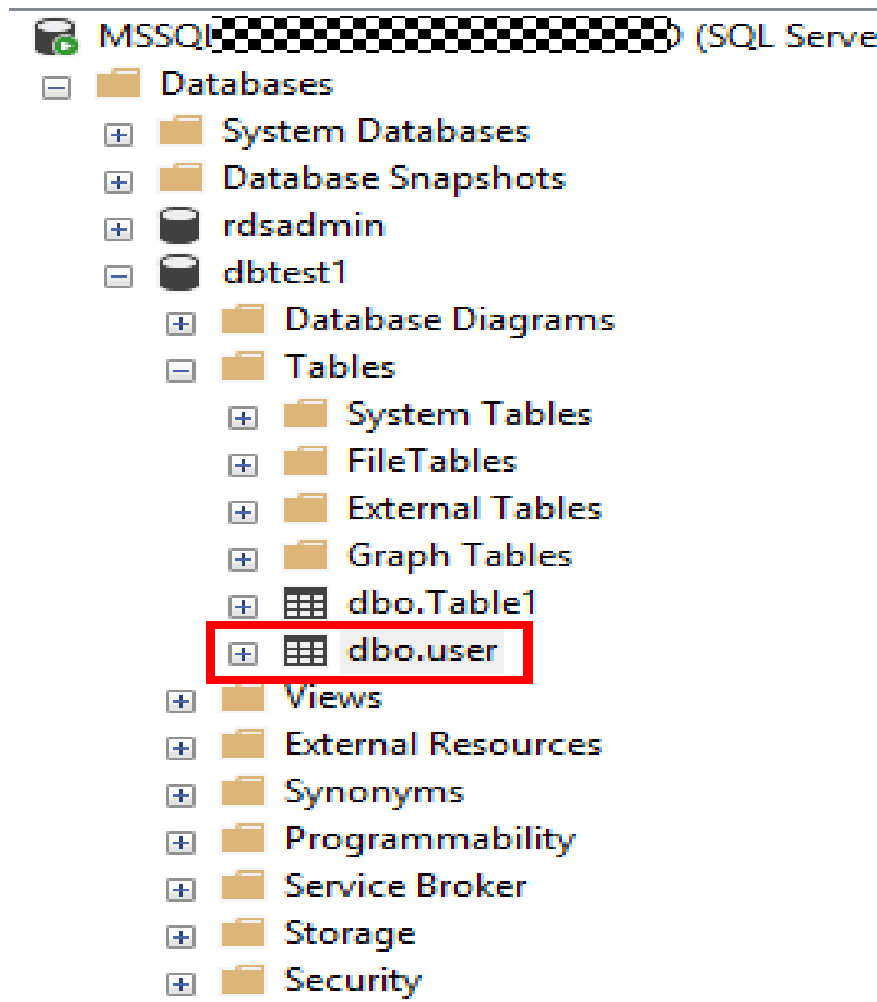
```
USE [master]
GO
EXEC master.dbo.sp_addlinkedserver @server = N'TEST_SERVERNAME', @srvproduct=N'SQLServer',
@provider=N'SQLOLEDB', @datasrc=N'192.168.***.***,1433'
EXEC master.dbo.sp_addlinkedsrvlogin @rmtsrsvname = N'TEST_SERVERNAME', @locallogin = NULL ,
@useself = N'false', @rmtuser = N'rdsuser', @rmtpassword = N'*****'
GO
```

Table 4-8 Parameter description

Parameter	Description
@server	The name of the linked server.
@srvproduct	The product name of the data source. Use the default value SQL Server .
@provider	Use the default value.
@datasrc	The IP address and port of the DB instance to be accessed.
@rmtsrsvname	The name of the linked server.
@locallogin	The login name on the local server. Use the default value NULL .
@useself	Whether to connect to the linked server by simulating the local login name or username and password. Enter false , which indicates that the linked server is connected through the username and password.
@rmtuser	The username (rdsuser).

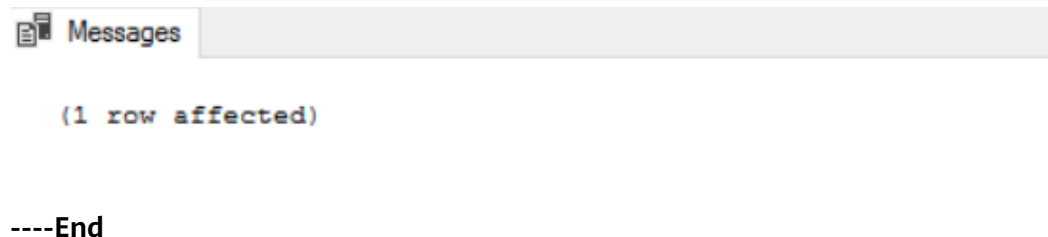
Parameter	Description
@rmtpassword	The password for the username.

Step 4 After the Dblink is created, you can view the databases created in DB instance 1 on the linked server.



Step 5 Run the following commands to check whether the data is successfully inserted, as shown in [Figure 4-21](#):

```
begin tran
set xact_abort on
INSERT INTO [LYNTEST].[dbtest1].[dbo].[user1]
([id],[lname],[rname])
VALUES('19','w','x')
GO
commit tran
```


Figure 4-21 Insert result

4.11 Deploying SQL Server Reporting Services (SSRS) on RDS for SQL Server

You can use SSRS to make various simple or complex reports. In addition, RDS provides the subscription function for you to subscribe to reports. This section describes how to deploy SSRS on RDS for SQL Server.

Scenarios

Microsoft SQL Server contains server components such as the SQL Server database engine, SSRS, and SQL Server Analysis Services (SSAS). The SQL Server database engine is a standard relational database component. RDS for SQL Server is a Platform-as-a-Service (PaaS) service that provides this database engine. However, other components, such as SSRS, are not provided as PaaS services on Huawei Cloud. To use SSRS on Huawei Cloud, you need to create a Windows-based ECS before installing and configuring SSRS.

SSRS has been separated from the Microsoft SQL Server component package and become an independent component service since SQL Server 2017. To migrate SSRS to the cloud, download the component from the Microsoft official website, install it on a Windows-based ECS, and use RDS for SQL Server as the backend database.

Prerequisites

- You have [created an RDS for SQL Server instance](#).
- You have created a Windows-based ECS. (The ECS and RDS DB instance must be in the same VPC, security group, and subnet.)

Procedure

- Step 1** Download [SSRS](#) and install it on the ECS.
- Step 2** After the installation is complete, click **Configure Report Server**.
- Step 3** In Report Server Configuration Manager, configure **Server Name** and click **Connect**.
- Step 4** In the navigation pane on the left, click **Service Account** and **Web Service URL** and configure parameters based on your service requirements.

NOTE

For details, see the [official documentation](#).

Step 5 Configure the report server.

1. In the navigation pane on the left, click **Database**. On the right of the page, click **Change Database** to create a report server database on the ECS.
2. In the displayed dialog box, select **Create a new report server database** and click **Next**.
If a local report database is available, you can use Data Replication Service (DRS) to **migrate the full backup files** of the local report database to the RDS for SQL Server instance.
3. Configure the connection information of the RDS for SQL Server instance. Set **Server Name** to the RDS for SQL Server instance address in the format of *IP address,port*. Use a comma (,) to separate the IP address and port. Set **Username** to **rdsuser**. Click **Test Connection**. After the connection test is successful, click **Next**.
4. Enter the database name, select a language for the script, and then click **Next**.
5. Configure the credentials for the account **rdsuser** to connect to the report server and click **Next**.
6. Confirm the report server information and click **Next**.
7. After the configuration is successful, click **Finish**.

 **NOTE**

For details, see the [official documentation](#).

Step 6 In the navigation pane on the left, click **Web Portal URL** and click **Apply**. After the operation is complete, click the URL to access the web page of the report server.

Step 7 In the upper right corner, choose **New > Data Source**.

Step 8 Configure the parameters as follows:

Table 4-9 Parameter description

Category	Parameter	Description
Properties	Name	Name of the data source. The name cannot contain the following characters: / @ \$ & * + = < > : ' , ? \
	Description	Description of the data source, which is used to identify different data sources.
	Hide	If this parameter is selected, the data source is hidden.
	Enable	If this parameter is selected, the data source is enabled.

Category	Parameter	Description
Connections	Type	Type of the data source. Select Microsoft SQL Server .
	Connection String	Domain name and database name of the RDS for SQL Server instance in the following format: Data Source=<Floating IP address of the RDS for SQL Server instance, port of the RDS for SQL Server instance>; Initial Catalog=<Database name>
Login	Data Source Login	Select Use the following credentials .
	Credential Type	Select Database username and password .
	Username	Account of the RDS for SQL Server instance
	Password	Password of the database account

Step 9 Click **Test Connection**. After the connection test is successful, click **Create**.

Step 10 After the data source is created, design reports using Report Builder or Visual Studio.

For details, see [Report Builder in SQL Server](#).

----End

4.12 Shrinking an RDS for SQL Server Database

Scenarios

You can use a stored procedure to shrink the size of the data and log files in a specified RDS for SQL Server database.

Prerequisites

An RDS for SQL Server DB instance has been connected. Connect to the DB instance through the Microsoft SQL Server client. For details, see [Connecting to a DB Instance Through a Public Network](#).

Constraints

- The database can be shrunk only when the database file size exceeds 50 MB. Otherwise, the following message is displayed:

```
Cannot shrink file '2' in database 'master' to 6400 pages as it only contains 2
```

- Transactions running at the row version control-based isolation level may prevent shrinking operations. To solve this problem, perform the following steps:

- a. Terminate the transactions that prevent shrinking.
- b. Terminate the shrinking operation. All completed tasks will be retained.
- c. Do not perform any operations and wait until the blocking transactions are complete.

Best Practices

When you plan to shrink a database, consider the following:

- A shrink operation is most effective after an operation that creates lots of unused space, such as a database reboot.
- Most databases require some free space to be available for regular day-to-day operations. If you shrink a database repeatedly and notice that the database size grows again, this indicates that the space that was shrunk is required for regular operations. In these cases, repeatedly shrinking the database is a wasted operation.
- A shrink operation does not preserve the fragmentation state of indexes in the database, and generally increases fragmentation to a degree. This is another reason not to repeatedly shrink the database.

Procedure

Step 1 Run the following command to shrink a database:

```
EXEC [master].[dbo].[rds_shrink_database] @DBName='myDbName';
```

Table 4-10 Parameter description

Parameter	Description
myDbName	Name of the database to be shrunk. If this parameter is not specified, all databases are shrunk by default.

The following figure shows the execution result set. Each result corresponds to the information about each file in the specified database (or all databases).

Figure 4-22 Result set

	DbId	FileId	CurrentSize	MinimumSize	UsedPages	EstimatedPages
1	1	1	688	512	512	512

Table 4-11 Result set parameter description

Column Name	Description
DbId	Database ID of the current shrink file.
FileId	File ID of the current shrink file.

Column Name	Description
CurrentSize	Number of 8 KB pages occupied by the file.
MinimumSize	Minimum number of 8 KB pages occupied by the file. The value indicates the minimum size or the initial size of the file.
UsedPages	Number of 8 KB pages used by the file.
EstimatedPages	Number of 8 KB pages that the database engine estimates the file can be shrunk to.

Step 2 After the command is successfully executed, the following information is displayed:

```
HW_RDS_Process_Successful: Shrink Database Done.
```

----End

Fault Rectification

If the file size does not change after the database is shrunk, run the following SQL statement to check whether the file has sufficient available space:

```
SELECT name, size/128.0 - CAST(FILEPROPERTY(name, 'SpaceUsed') AS int)/128.0 AS AvailableSpaceInMB FROM sys.database_files;
```

Example

- Run the following command to shrink the **dbtest2** database:
EXEC [master].[dbo].[rds_shrink_database] @DBName = 'dbtest2';
The command output is as follows.

Figure 4-23 Execution result

```
[Shrink Start] Date and time: 2020-03-19 15:51:07

Start to shrink files in database [dbtest2], current file id is 1...
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
Shrink file (id: 1) in database [dbtest2] done!

Start to shrink files in database [dbtest2], current file id is 2...
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
Shrink file (id: 2) in database [dbtest2] done!

[Shrink End] Date and time: 2020-03-19 15:51:08

HW_RDS_Process_Successful : Shrink Database done.
```

- Run the following command to shrink all databases:
EXEC [master].[dbo].[rds_shrink_database];


4.13 Using DAS to Create and Configure Agent Job and Dblink on the Master and Slave Databases for RDS for SQL Server Instances


Scenarios

Data Admin Service (DAS) is a one-stop database management platform that allows you to manage databases on a web console. It offers database development, O&M, and intelligent diagnosis, facilitating your database usage and maintenance. Currently, DAS supports primary/standby switchover of RDS for SQL Server databases, facilitating synchronization between master and slave databases.

Logging In to DAS

Step 1 [Log in to the management console](#).

Step 2 Click  in the upper left corner and select a region.

Step 3 Click  in the upper left corner of the page and choose **Databases > Relational Database Service**.

Step 4 On the **Instances** page, locate the target DB instance and click **Log In** in the **Operation** column.

Alternatively, click the instance name on the **Instances** page. On the displayed **Basic Information** page, click **Log In** in the upper right corner of the page.

Step 5 On the displayed login page, enter the correct username and password and click **Log In**.

----End

Creating a Job for Data Synchronization to the Slave Database

Step 1 Create a job on the master database.

On the DAS console, choose **SQL Operations > SQL Query** on the top menu bar. In the msdb database, run the following commands to create a job:

NOTE

If a job has been created on the master database, skip this step.

```
USE [msdb]
GO
DECLARE @jobId BINARY(16)
EXEC msdb.dbo.sp_add_job @job_name=N'hwtest',
    @enabled=1,
    @notify_level_eventlog=0,
    @notify_level_email=2,
    @notify_level_page=2,
    @delete_level=0,
```

```

        @category_name=N'[Uncategorized (Local)]',
        @owner_login_name=N'rdsuser', @job_id = @jobId OUTPUT
select @jobId
GO
EXEC msdb.dbo.sp_add_jobserver @job_name=N'hwtest', @server_name = N'*****'
GO
USE [msdb]
GO
EXEC msdb.dbo.sp_add_jobstep @job_name=N'hwtest', @step_name=N'select orders',
        @step_id=1,
        @cmdexec_success_code=0,
        @on_success_action=1,
        @on_fail_action=2,
        @retry_attempts=0,
        @retry_interval=0,
        @os_run_priority=0, @subsystem=N'TSQL',
        @command=N'select * from orders;',
        @database_name=N'test',
        @flags=0
GO
USE [msdb]
GO
EXEC msdb.dbo.sp_update_job @job_name=N'hwtest',
        @enabled=1,
        @start_step_id=1,
        @notify_level_eventlog=0,
        @notify_level_email=2,
        @notify_level_page=2,
        @delete_level=0,
        @description=N'',
        @category_name=N'[Uncategorized (Local)]',
        @owner_login_name=N'zf1',
        @notify_email_operator_name=N'',
        @notify_page_operator_name=N''
GO

```

Run the following SQL statements to check whether the job has been created:

use [msdb]

select * from msdb.dbo.sysjobs where name ='hwtest';

Step 2 Switch to the slave database.

 **NOTE**

Currently, RDS for SQL Server does not support job synchronization between the master and slave databases. Therefore, you need to create a job on the slave database and synchronize the job. Click [Switch SQL Execution Node](#) next to **Master** to switch to the slave database.

Step 3 Run the commands in [Step 1](#) to create a job on the slave database.

Alternatively, use SQL Server Management Studio (SSMS) to export the created job to the editor window, copy the job to the SQL query window, and then run the commands in [Step 1](#) to create a job on the slave database.

If the job fails to be created, you are advised to delete the job first and then create the job again.

Figure 4-24 Exporting a job

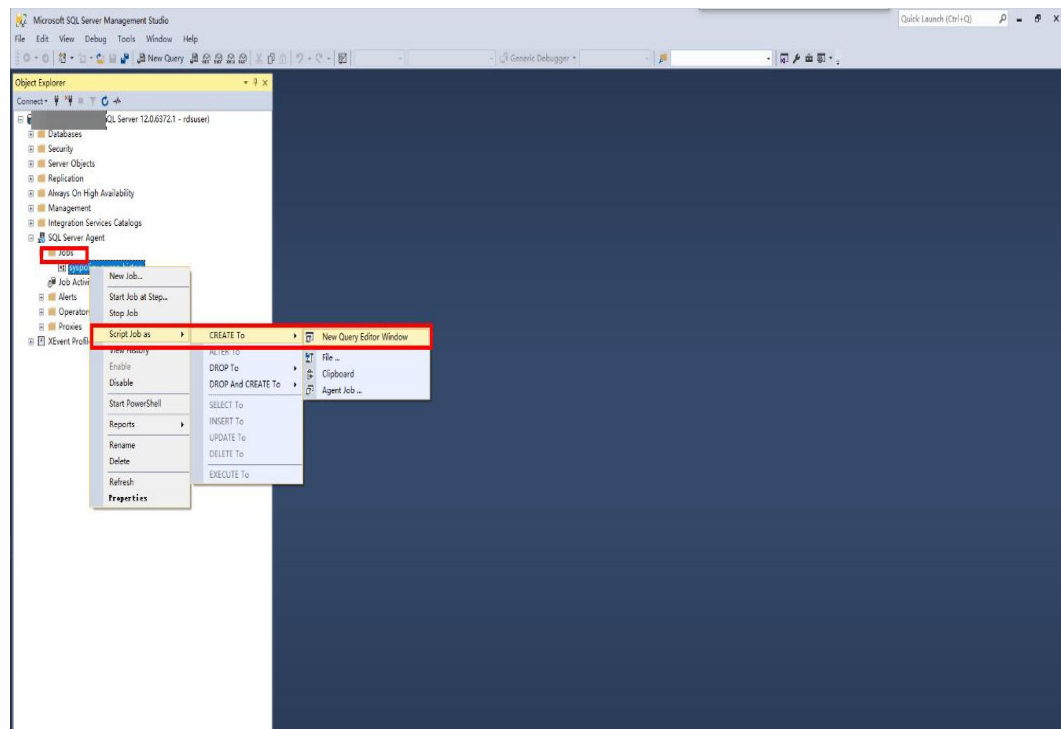
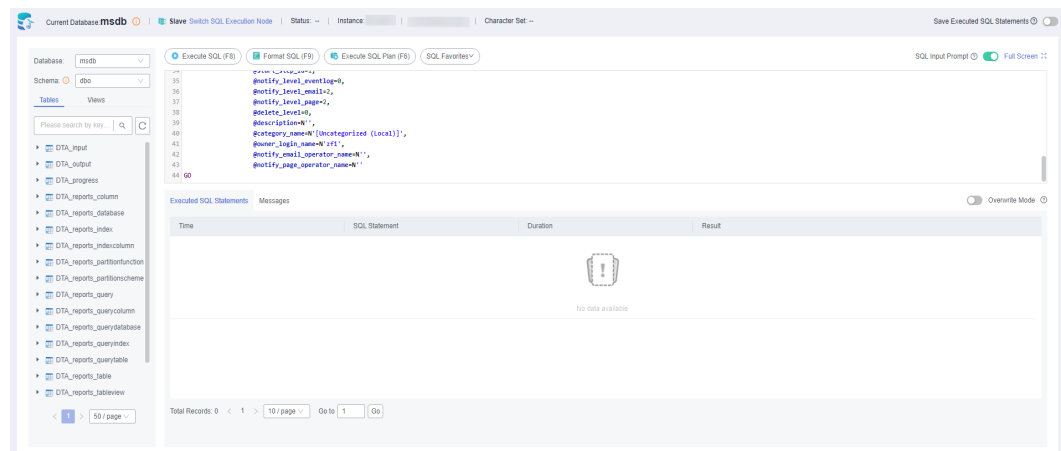


Figure 4-25 Using the DAS console to create a job on the slave database



Run the following SQL statements to delete the job:

USE [msdb]

GO

**EXEC msdb.dbo.sp_delete_job @job_name=N'hwtest',
@delete_unused_schedule=1**

GO

----End

Creating a Dblink for Data Synchronization to the Slave Database

DAS allows you to create linked servers to synchronize data between master and slave databases.

NOTE

Check whether the MSDTC is configured by referring to [Creating a Linked Server for an RDS for SQL Server DB Instance](#).

Step 1 Create a Dblink on the master database.

USE [master]

GO

```
EXEC master.dbo.sp_addlinkedserver @server = N'TEST',
@srvproduct=N'mytest', @provider=N'SQLOLEDB', @datasrc=N'abcd'
```

```
EXEC master.dbo.sp_addlinkedsrvlogin @rmtsrvname = N'TEST', @locallogin =
NULL , @useself = N'False', @rmtuser = N'rdsuser', @rmtpassword = N'*****'
```

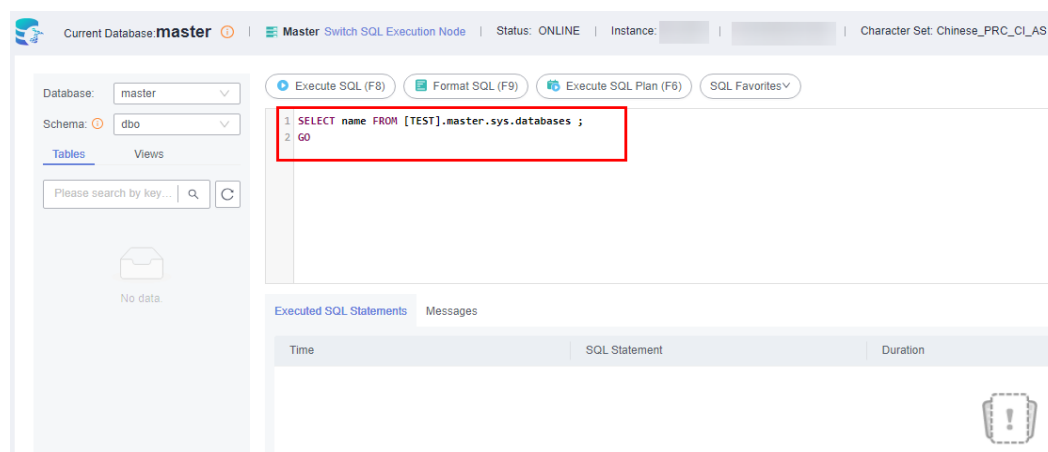
GO

After the creation is successful, the corresponding DB instance or databases can be linked to view data verification. Run the following statements to query databases:

```
SELECT name FROM [TEST].master.sys.databases ;
```

GO

Figure 4-26 Database query



Step 2 Create a Dblink on the slave database.

On the DAS console, click **Switch SQL Execution Node** next to **Master** and run the SQL statement for creating a Dblink.

NOTE

If the current DB instance and the interconnected database are not in the same VPC or distributed transactions are enabled with an EIP bound to the primary DB instance, the query statement cannot be executed on the standby DB instance. This step is used only to synchronize the DBLink configuration. After a switchover or failover, the DBLink can be used.

----End

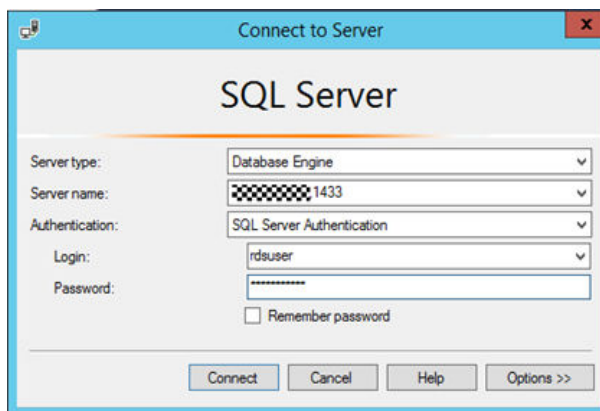
4.14 Creating a Job for Scheduled Instance Maintenance

Scenarios

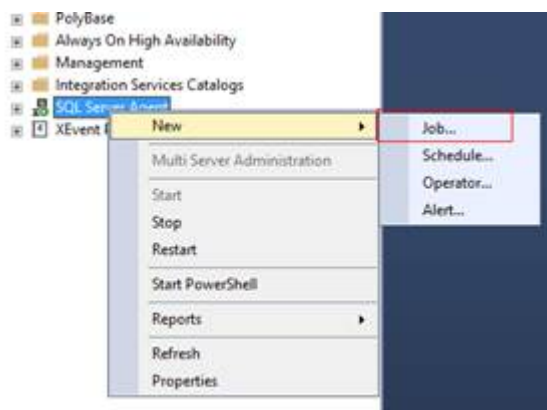
After a DB instance runs for a period of time, the system performance deteriorates because index fragments increase and statistics are not updated in a timely manner. You are advised to create a SQL agent job to periodically re-create indexes, update statistics, and shrink the database.

Creating an Index Re-creating Job

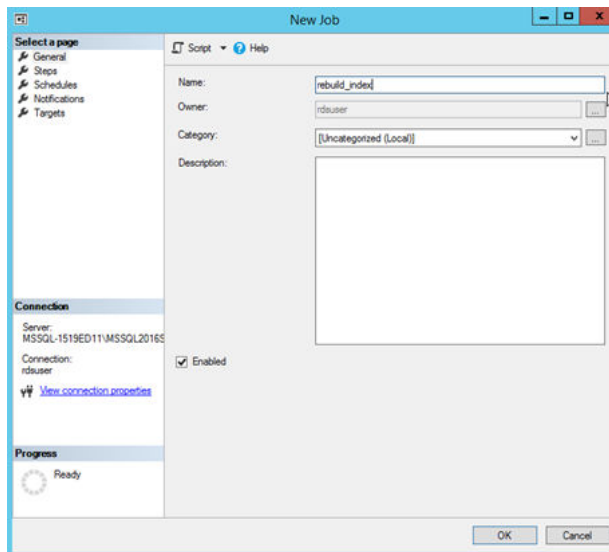
Step 1 Start the SQL Server Management Studio client and log in to it as user **rduser**.



Step 2 Right-click **SQL Server Agent** and choose **New > Job** to create an SQL agent job.

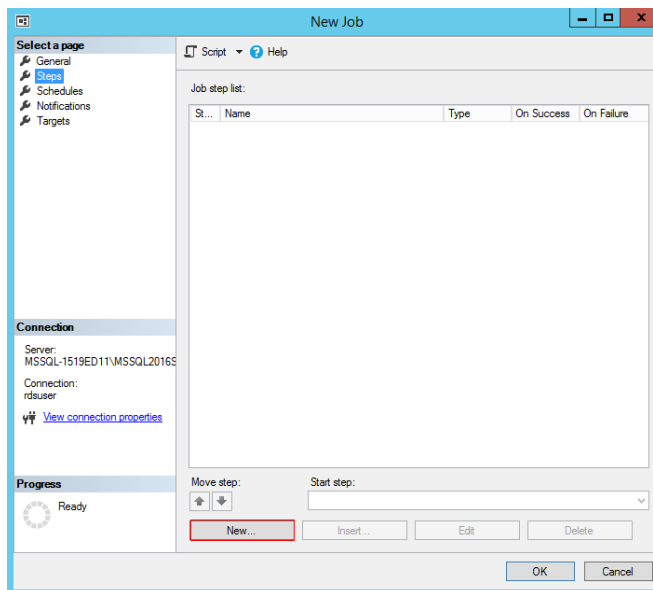


Step 3 Enter the name and description, and click **OK**.



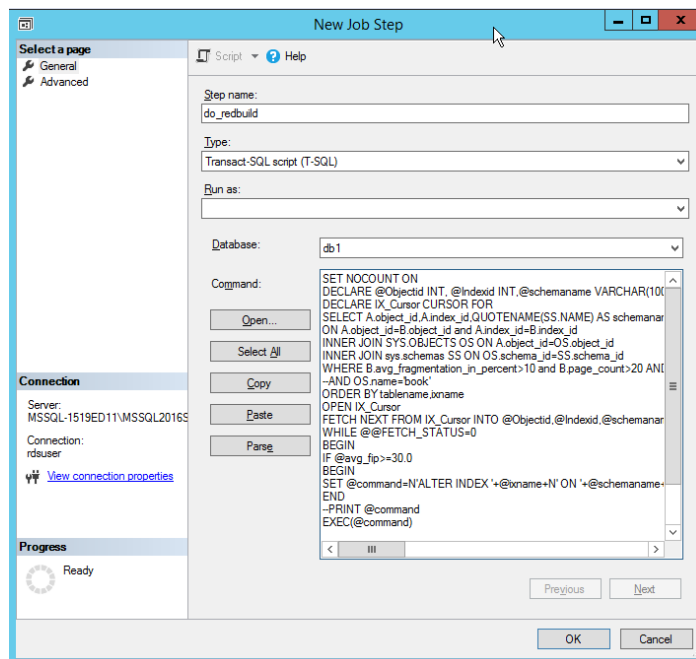
Step 4 Select **Steps** and click **New** to add an execution step.

Figure 4-27 Adding an execution step



Step 5 Enter the step name, type, and command, and click **OK**. Set **Command** to the SQL statements that need to be executed periodically. When the number of index fragments reaches a specified value, for example, 30%, the index can be recreated.

Figure 4-28 Configure parameters



Run the following SQL statement to recreate the index because the number of index fragments of all tables in the specified **dbname** exceeds 30%:

```

use [dbname]
SET NOCOUNT ON
DECLARE @Objectid INT, @Indexid INT,@schemaname VARCHAR(100),@tablename
VARCHAR(300),@ixname VARCHAR(500),@avg_fip float,@command VARCHAR(4000)
DECLARE IX_Cursor CURSOR FOR
SELECT A.object_id,A.index_id,QUOTENAME(SS.name) AS
schemaname,QUOTENAME(OBJECT_NAME(B.object_id,B.database_id))as
tablename ,QUOTENAME(A.name) AS ixname,B.avg_fragmentation_in_percent AS avg_fip FROM
sys.indexes A inner join sys.dm_db_index_physical_stats(DB_ID(),NULL,NULL,NULL,'LIMITED') AS B
ON A.object_id=B.object_id and A.index_id=B.index_id
INNER JOIN sys.objects OS ON A.object_id=OS.object_id
INNER JOIN sys.schemas SS ON OS.schema_id=SS.schema_id
WHERE B.avg_fragmentation_in_percent>10 and B.page_count>20 AND A.index_id>0 AND A.is_disabled<>1
--AND OS.name='book'
ORDER BY tablename,ixname
OPEN IX_Cursor
FETCH NEXT FROM IX_Cursor INTO @Objectid,@Indexid,@schemaname,@tablename,@ixname,@avg_fip
WHILE @@FETCH_STATUS=0
BEGIN
IF @avg_fip>=30.0
BEGIN
SET @command=N'ALTER INDEX '+@ixname+N' ON '+@schemaname+N'.'+ @tablename+N' REBUILD ';
END
--PRINT @command
EXEC(@command)
FETCH NEXT FROM IX_Cursor INTO @Objectid,@Indexid,@schemaname,@tablename,@ixname,@avg_fip
END
CLOSE IX_Cursor
DEALLOCATE IX_Cursor
    
```

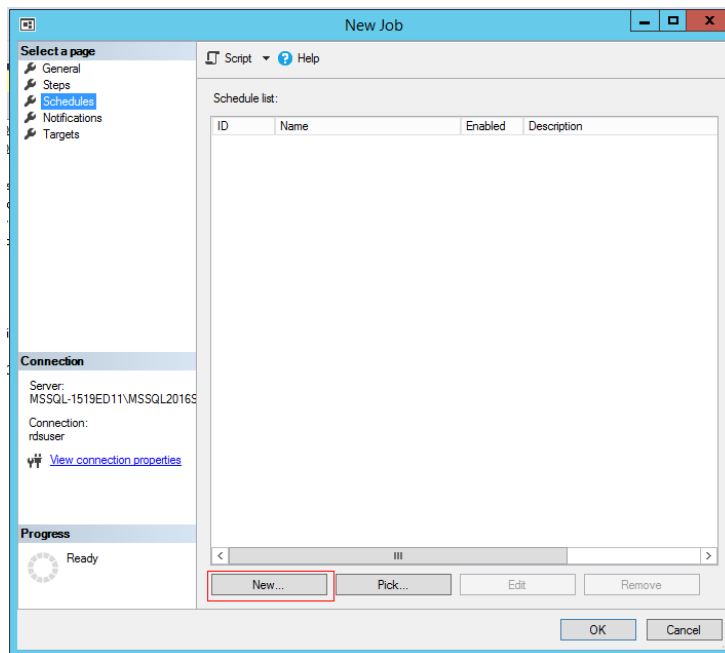
NOTE

In the preceding SQL statements, you only need to change the value of Use **[dbname]** in the first line to the specified database name.

If you need to execute the SQL statements for all databases, modify the SQL statements to add cyclic execution for all databases.

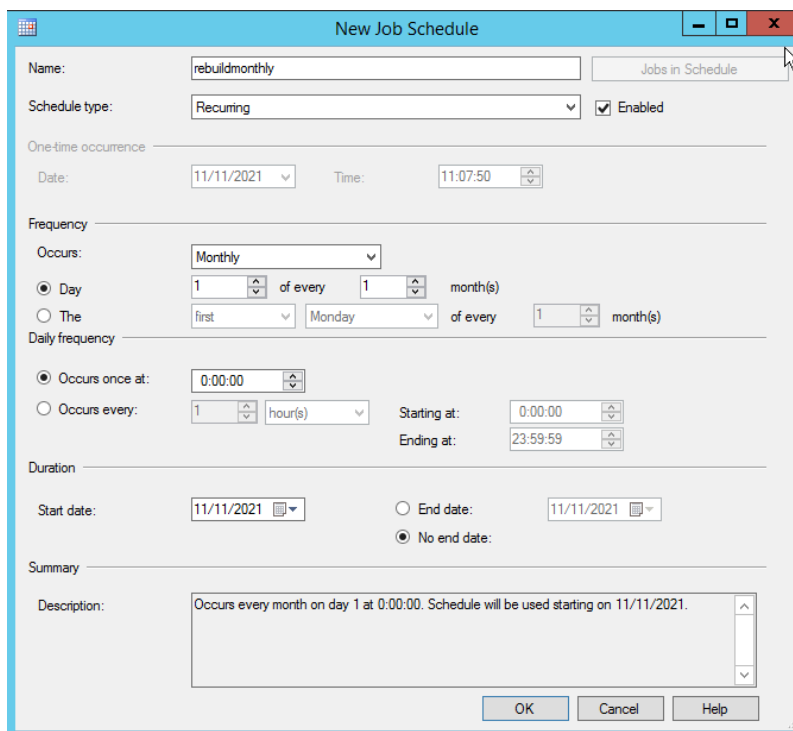
Step 6 Select **Schedules** and click **New** to add a scheduled execution plan.

Figure 4-29 Adding a scheduled execution plan



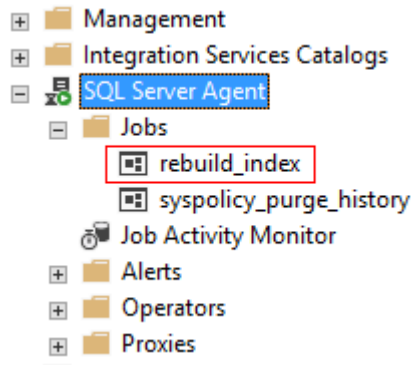
Step 7 Add a schedule that is executed once a month, modify the daily frequency and duration, and click **OK**.

Figure 4-30 Configuring a scheduled execution plan



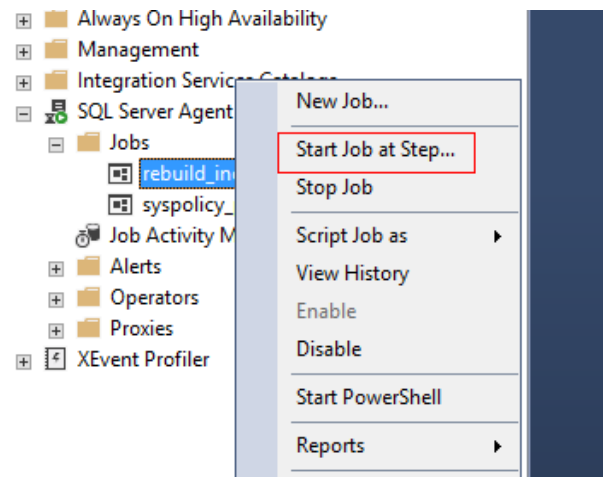
Step 8 View that the job has been created.

Figure 4-31 job

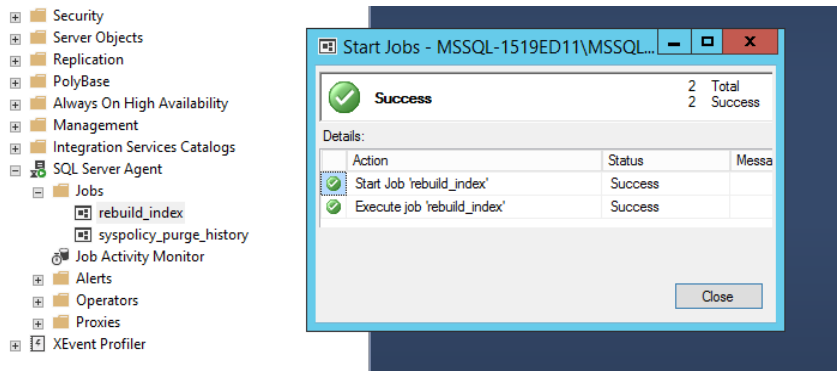


Step 9 Right-click the job and choose **Start Job at Step** to manually run the job.

Figure 4-32 Running a job.



Step 10 Check whether the job can run properly. If the job runs normally, the maintenance job for periodically recreating the indexes of the **db1** database has been created.



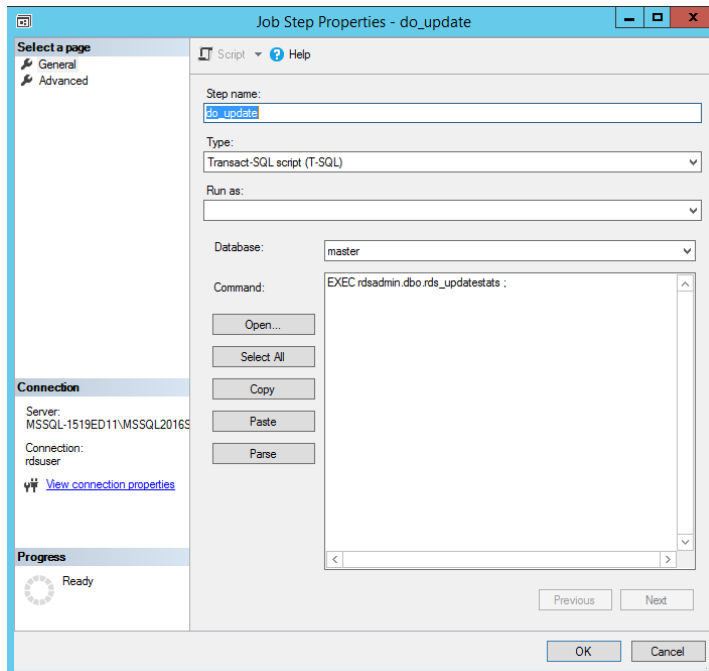
----End

Updating Statistics

Step 1 Perform [Step 1](#) to [Step 4](#).

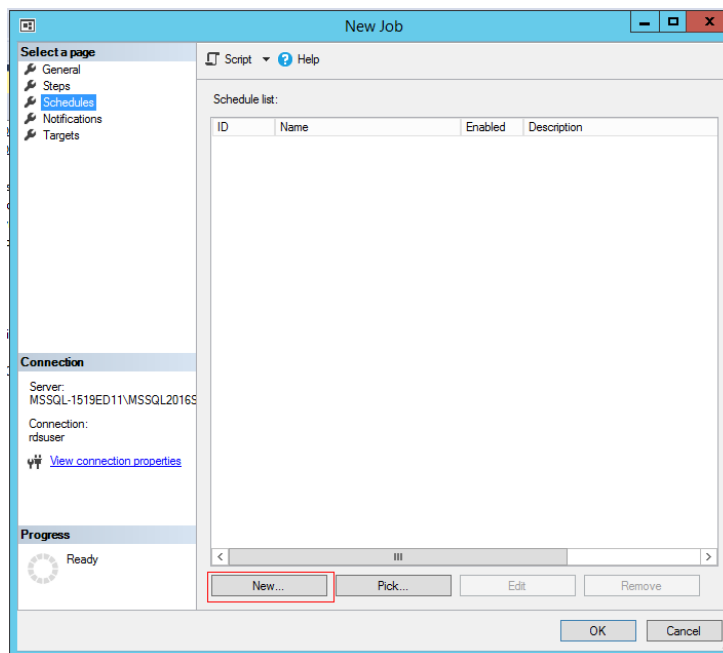
Step 2 Enter the step name, type, and command, and click **OK**. Set **Command** to the stored procedure for updating statistics. For details, see [Updating Database Statistics](#).

Figure 4-33 Updating statistics



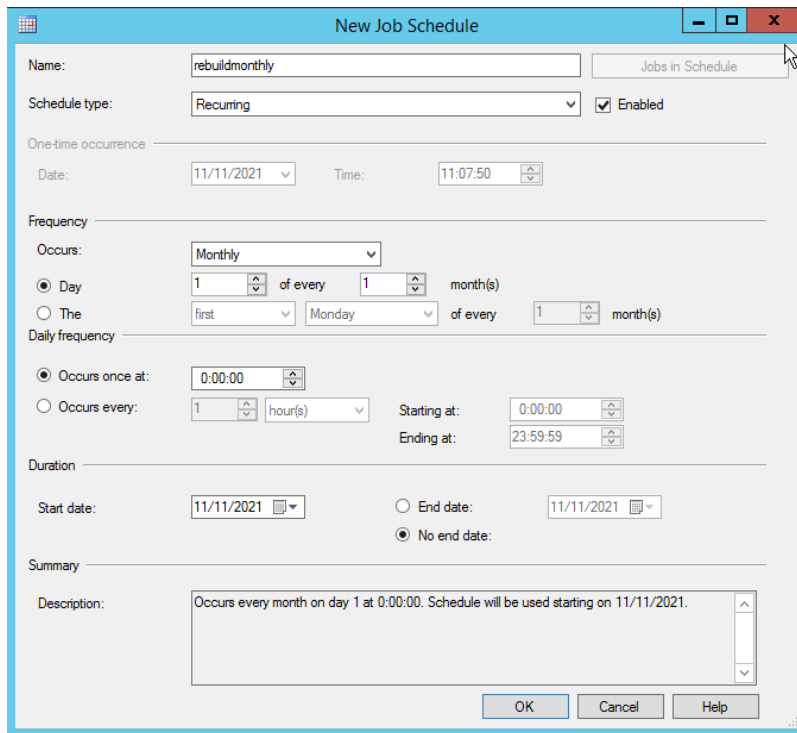
Step 3 Select **Schedules** and click **New** to add a scheduled execution plan.

Figure 4-34 Adding a scheduled execution plan



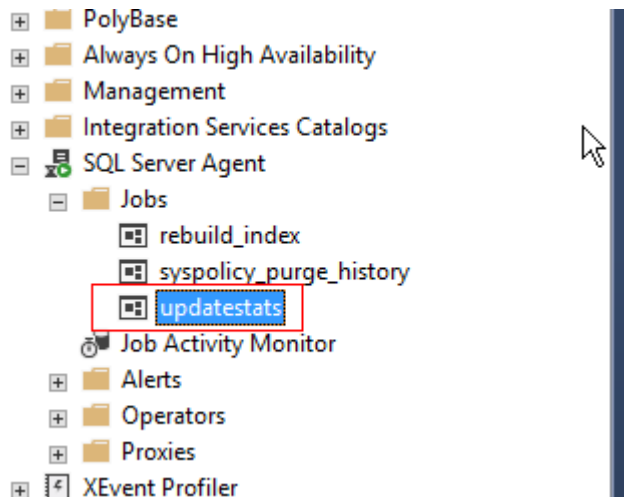
Step 4 Add a schedule that is executed once a month, modify the daily frequency and duration, and click **OK**.

Figure 4-35 Configuring a scheduled execution plan



Step 5 View that the job has been created.

Figure 4-36 Updating statistics job



Step 6 Right-click the job and choose **Start Job at Step** to manually run the job.

----End

Shrinking the Database Periodically

Step 1 Perform Step 1 to Step 4.

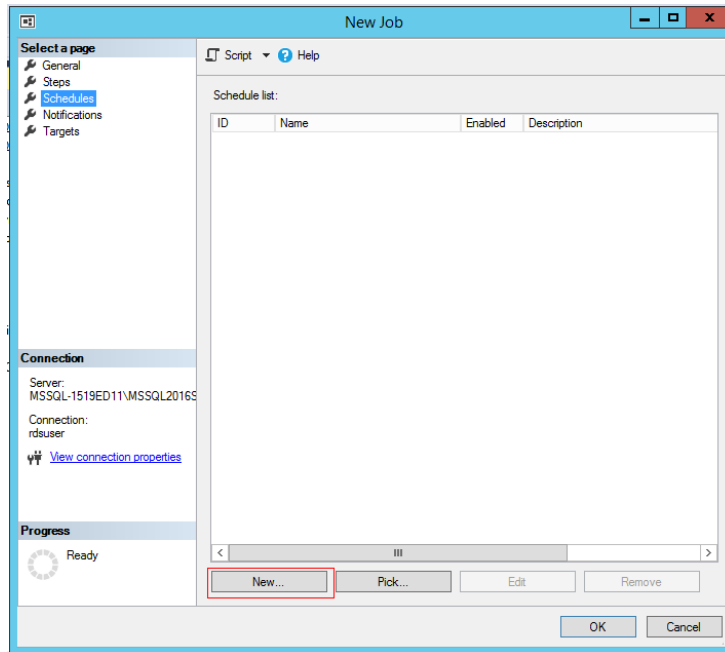
Step 2 Enter the step name, type, and command, and click **OK**. Set **Command** to the SQL commands for shrinking the database.

```
EXEC [master].[dbo].[rds_shrink_database_log] @dbname='myDbName';
```


Set **@dbname** to the database name.

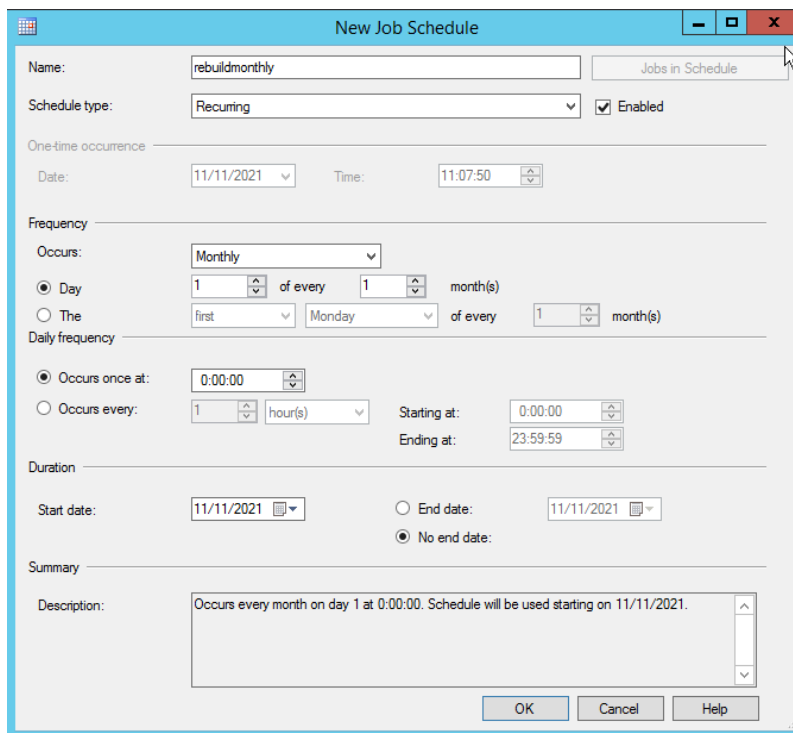
Step 3 Select **Schedules** and click **New** to add a scheduled execution plan.

Figure 4-37 Adding a scheduled execution plan



Step 4 Add a schedule that is executed once a month, modify the daily frequency and duration, and click **OK**.

Figure 4-38 Configuring a scheduled execution plan



Step 5 Right-click the job and choose **Start Job at Step** to manually run the job.

----End

4.15 Using Extended Events

Extended event permissions are available now. You can use **rdsuser** to manage extended events or grant extended event permissions to other users.

For more information, see [Quickstart: Extended Events in SQL Server](#).

Constraints

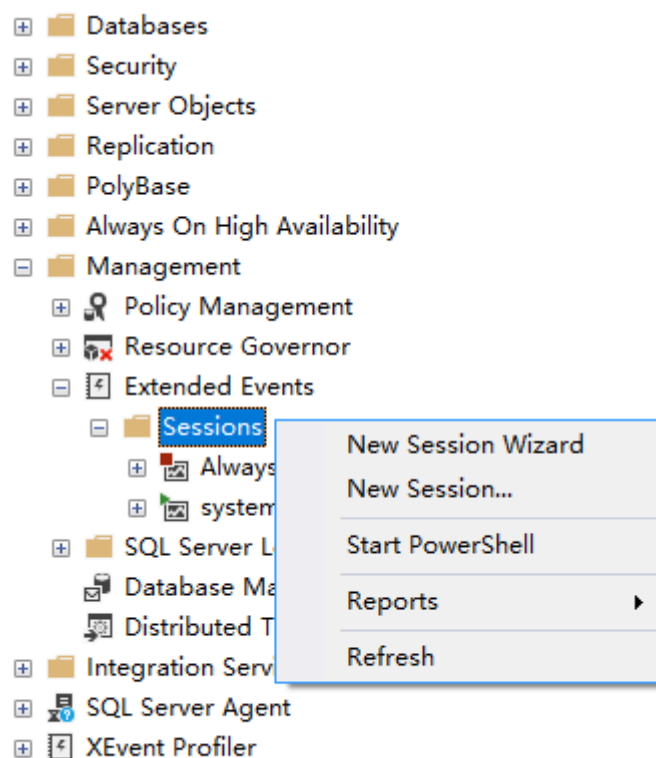
- All RDS for SQL Server 2008 versions do not support extended events because Microsoft SQL Server 2008 does not support extended events.
- The `etw_classic_sync_target` type is not available for extended event targets.
- When an extended event is created or updated, only the path `D:\RDSDBDATA\Log\error` is supported. The file name can be customized.

Creating an Extended Event

Step 1 Start the SQL Server Management Studio (SSMS) client and log in to it as user **rdsuser**.

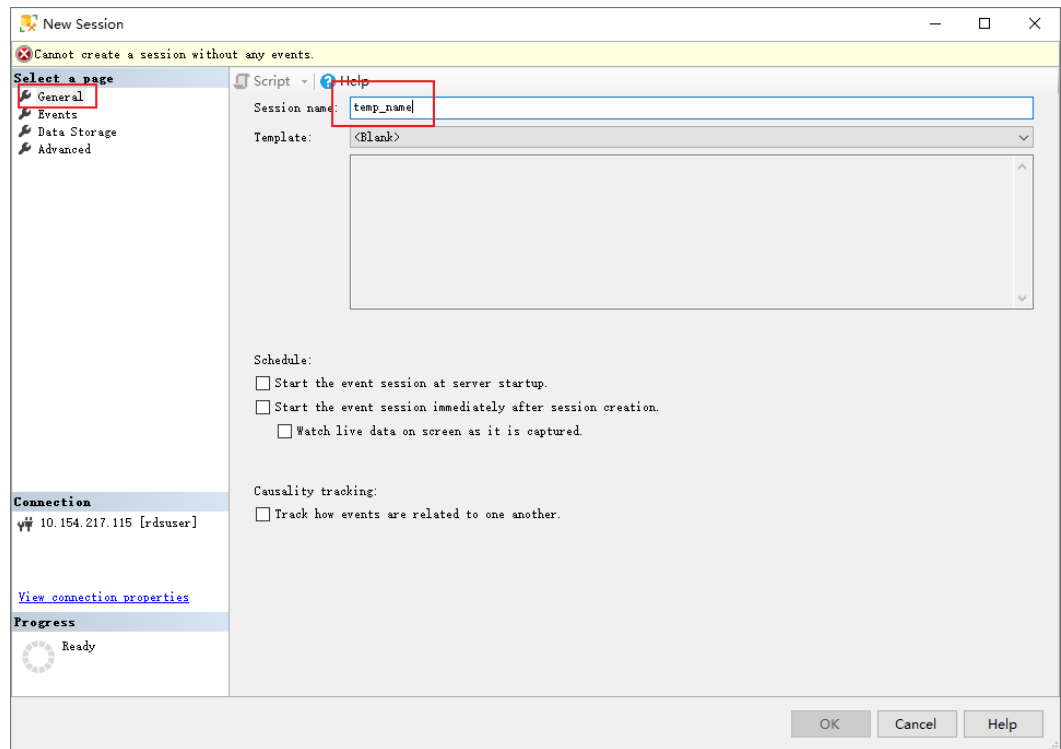
Step 2 Choose **Management > Sessions > New Session**.

Figure 4-39 Creating an extended event



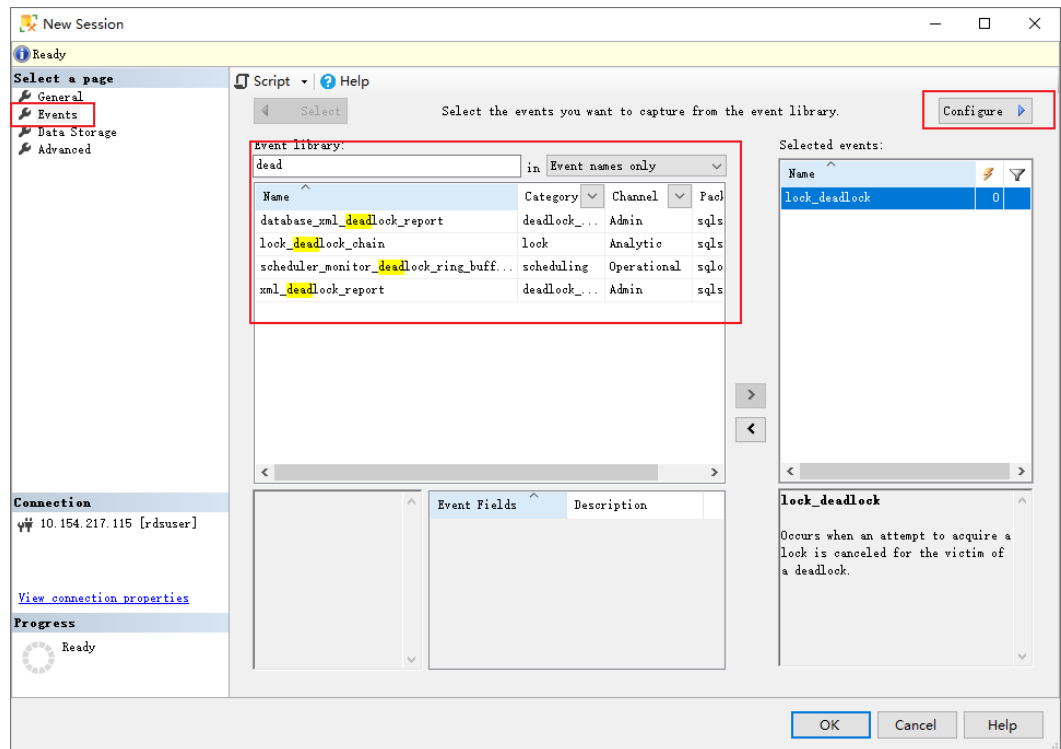
Step 3 Click **General** and enter a session name.

Figure 4-40 Entering a session name



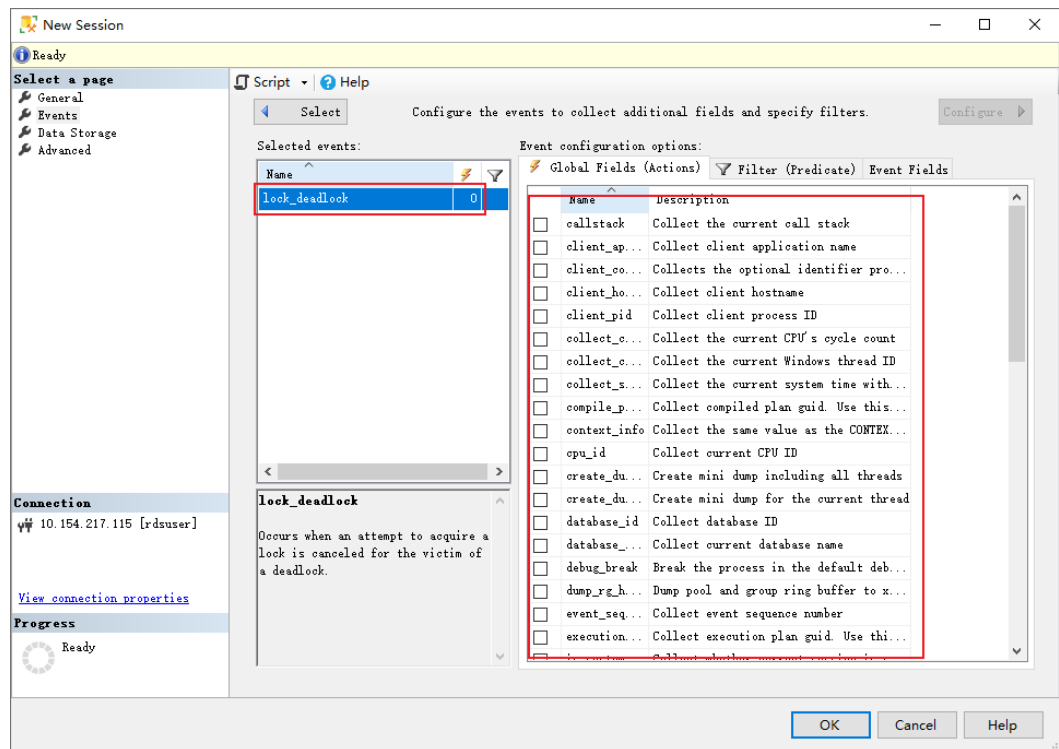
Step 4 Click **Events** and select an event.

Figure 4-41 Selecting an event



Step 5 Click **Configure** on the page displayed in **Step 4**.

Figure 4-42 Configuring an event

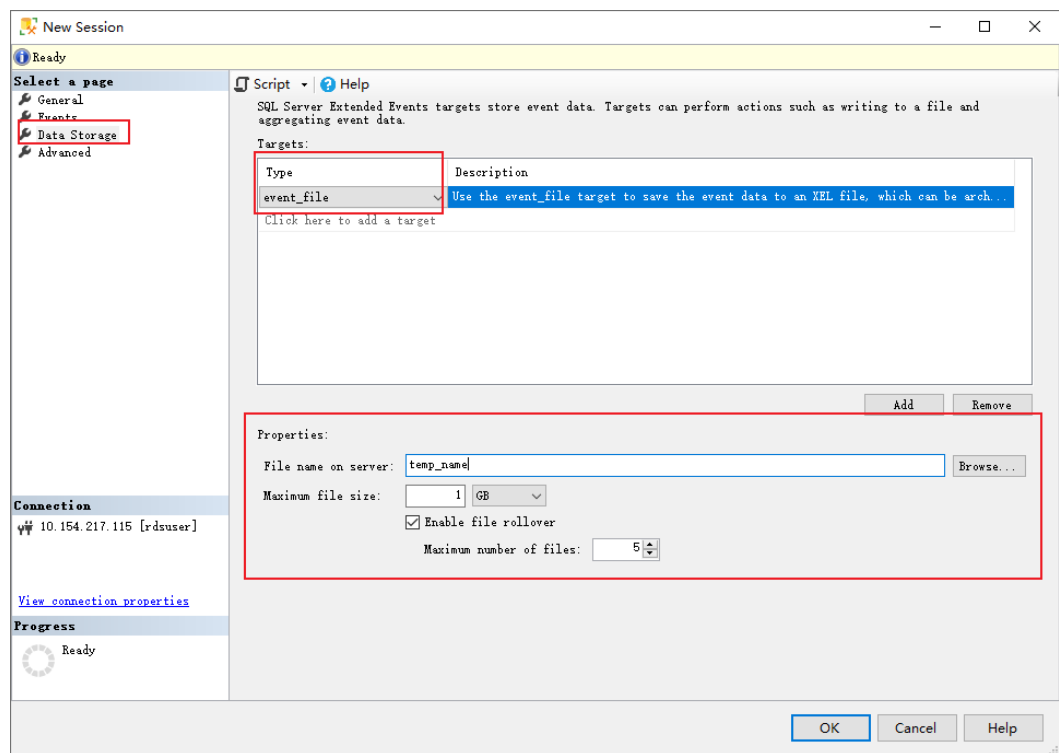


Step 6 Click **Data Storage** to configure the data storage.

NOTE

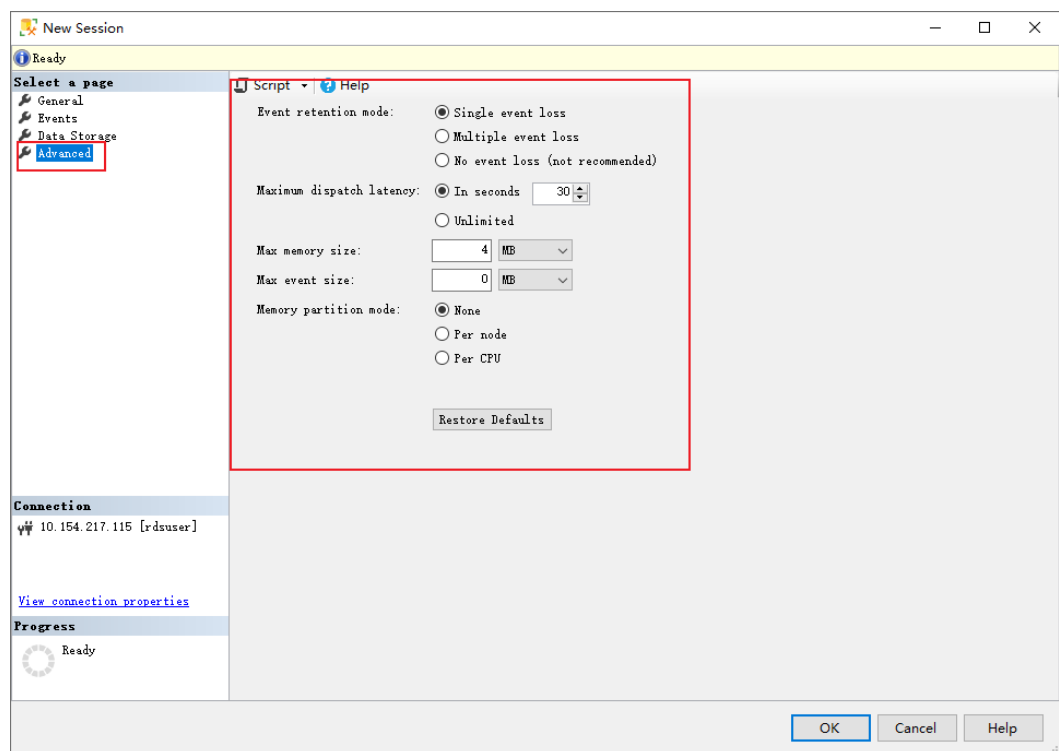
The file name can be customized. There is no need to click **Browse** because you can only browse the file system of the client where the SSMS is located, but not the file system of the RDS for SQL Server server. RDS for SQL Server supports the **D:\RDSDBDATA\Log\error** path only, so you only need to change the file name.

Figure 4-43 Configuring the data storage



Step 7 Click **Advanced** to configure the file generation policy.

Figure 4-44 Configuring the file generation policy



Step 8 Use the script to generate SQL statements. After confirming that the SQL statements are correct, run the SQL statements to create an extended event.

```
-- Example SQL statements generated
CREATE EVENT SESSION [temp_name] ON SERVER
ADD EVENT sqlserver.lock_deadlock(
ACTION(sqlserver.session_id,sqlserver.sql_text,sqlserver.username))
ADD TARGET package0.event_file(SET filename=N'temp_name')
GO
```

----**End**