RDS for MySQL

Best Practices

Issue	01
Date	2025-07-17

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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 Best Practices

This chapter describes best practices for working with RDS for MySQL and provides operational guidelines that you can follow when using this service.

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.
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.

Reference	Description
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.

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

2.1 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.

2.2 Resource Planning

Table 2-1 Resource planning description

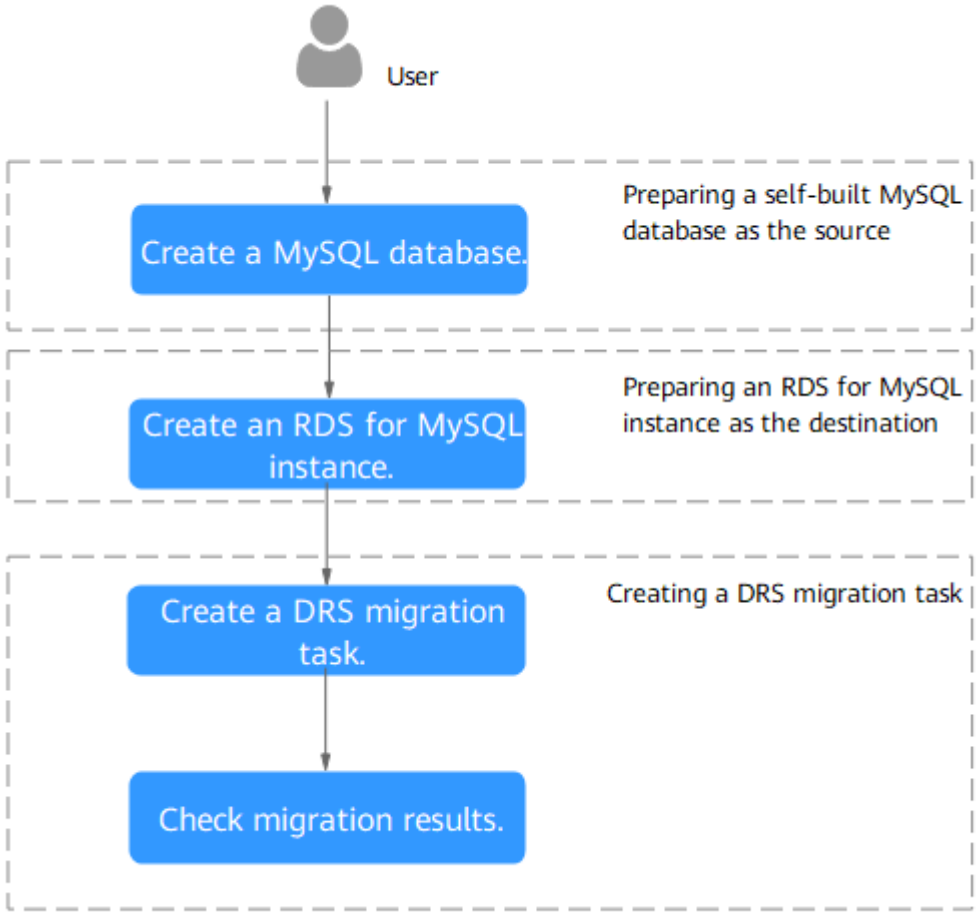
Category	Subcategory	Planned Value	Remarks
RDS	RDS instance name	rds-mysql	Customize a name for easy identification.
	DB engine version	MySQL 5.7	-
	Instance type	Single	In this practice, select a single instance. To improve service reliability, selecting a primary/standby instance is recommended.
	Storage 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.

2.3 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



2.4 Cloud Migration

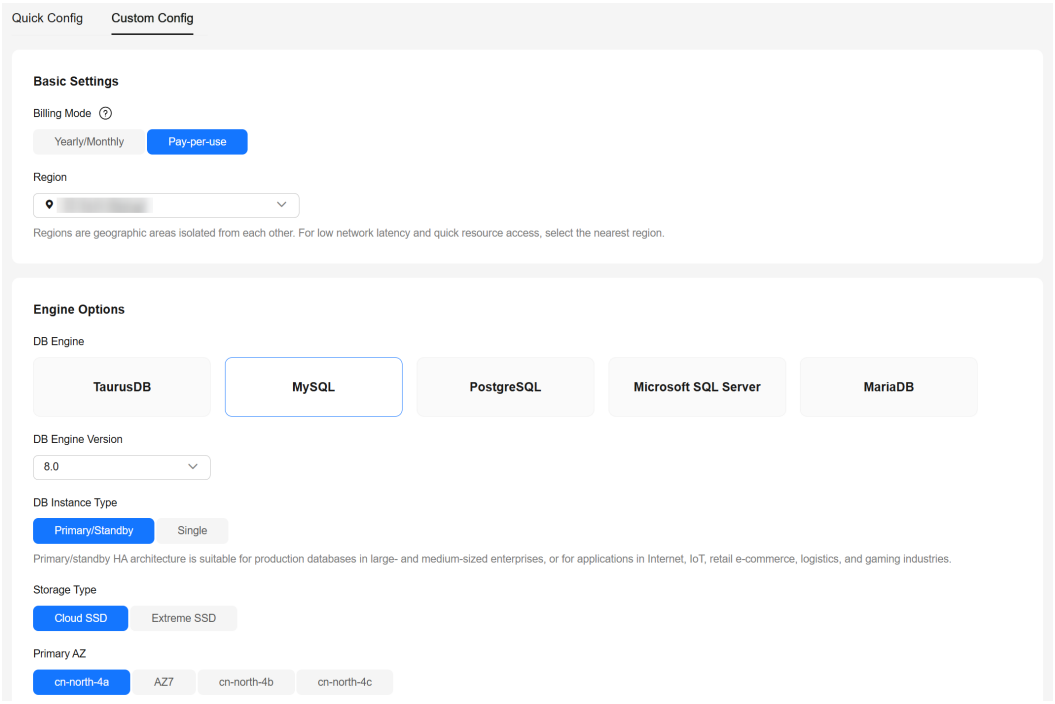
2.4.1 Creating an RDS for MySQL Instance

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.

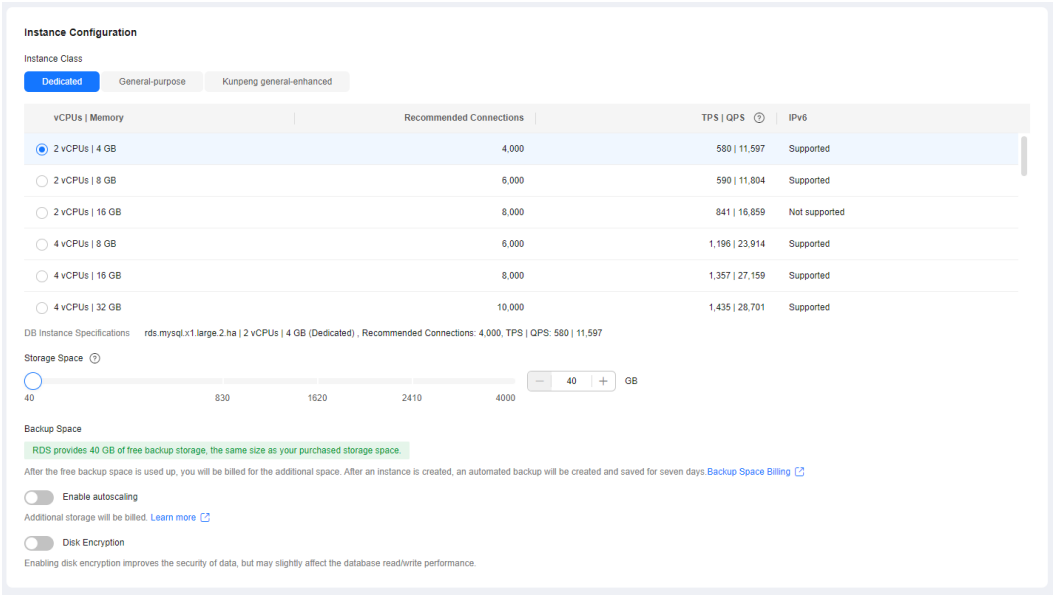
Step 2 Configure basic information for the instance. Select **CN-Hong Kong** for **Region**.

Figure 2-2 Basic information



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

2.4.2 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

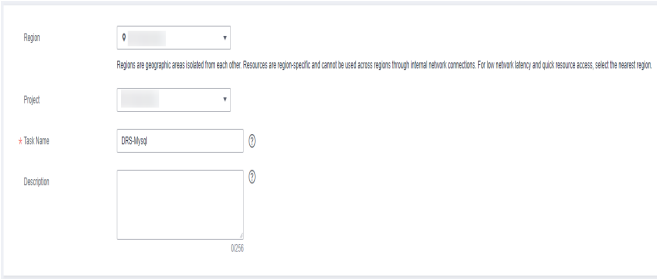
Migrate the **loadtest** database from a self-managed MySQL server to an RDS for MySQL instance.

Step 1 Go to the **Create Migration Task** page.

Step 2 Configure parameters as needed.

1. Specify a migration task name. Select the region where the target instance is located, for example, **CN-Hong Kong**.

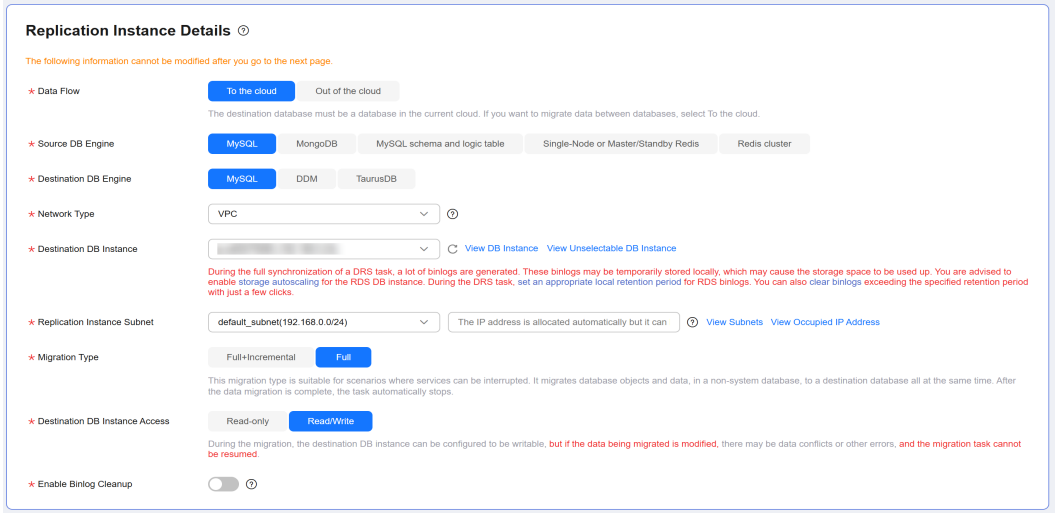
   Figure 2-4 Migration task

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**.

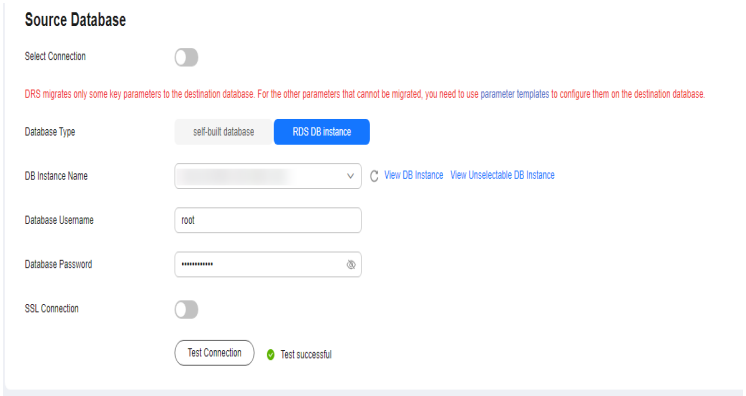
Step 3 Click **Create Now**.

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

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

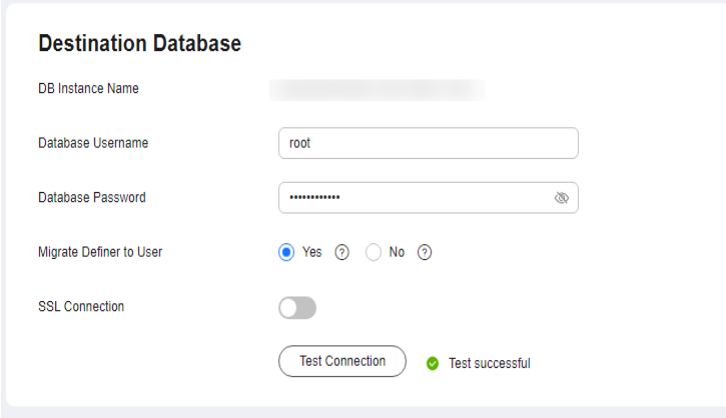
1. Configure source database information.
2. Click **Test Connection**.
   If a successful test message is returned, login to the source is successful.

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



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

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

For more information, see **From MySQL to MySQL**.

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

2.4.3 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**.

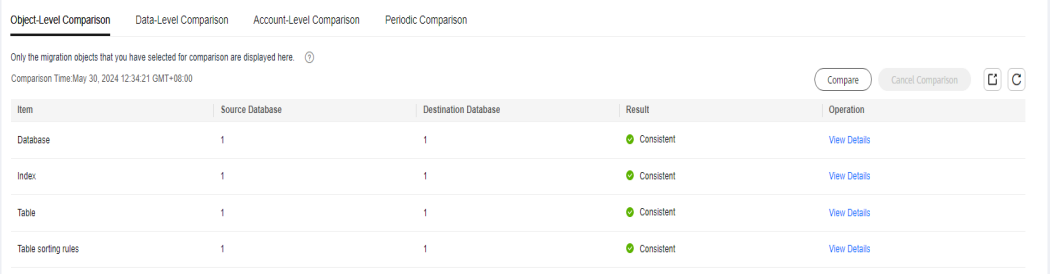
Step 3 Click the service list icon on the left and choose **Databases** > **Data Replication Service**.

Step 4 Click the name of the DRS migration task.

Step 5 In the navigation pane, choose **Migration Comparison**.

Step 6 By default, the **Object-Level Comparison** tab page is displayed. Click **Compare** and check the comparison results of the items such as databases, tables, and indexes between the source and destination databases.

Figure 2-8 Migration comparison



Step 7 Click the **Data-Level Comparison** tab, create a comparison task, and check the data comparison results between the source and destination databases.

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

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

3.1 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 greater than or equal to $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)**.

3.2 Resource Planning

Table 3-1 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	-
	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	-

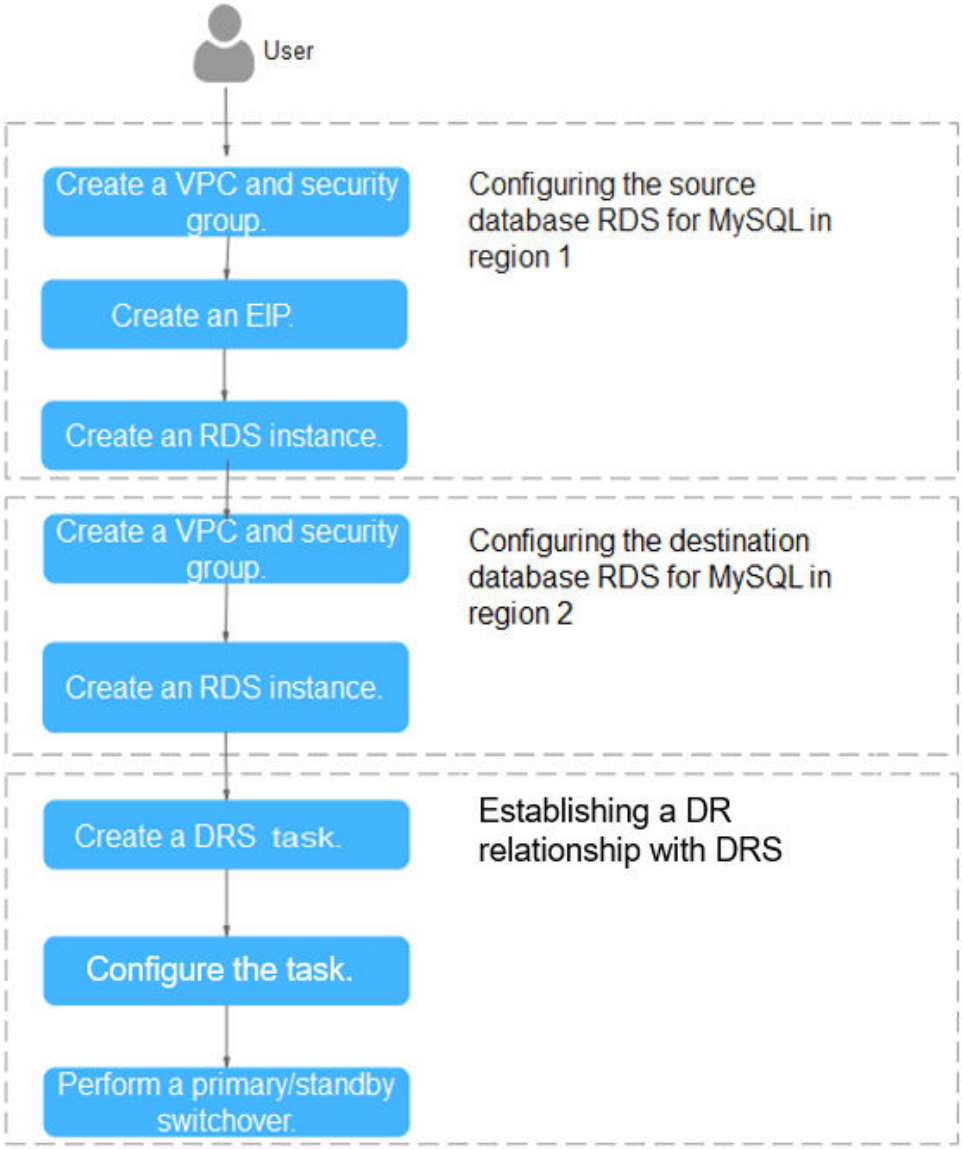
Category	Subcategory	Planned Value	Description
	Instance type	Single	A single instance is used in this example. To improve service reliability, select a primary/standby instance.
	Storage type	Ultra-high I/O	-
	AZ	AZ2	AZ2 is selected in this example. To improve service reliability, select the primary/standby instance type and deploy the primary and standby instances in different AZs.
	Instance specifications	General-enhanced 2 vCPUs | 4 GB	-
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.

Category	Subcategory	Planned Value	Description
	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.

3.3 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 3-1 Flowchart



3.4 Configuring an RDS for MySQL Instance in the Production Center

3.4.1 Creating a VPC and Security Group

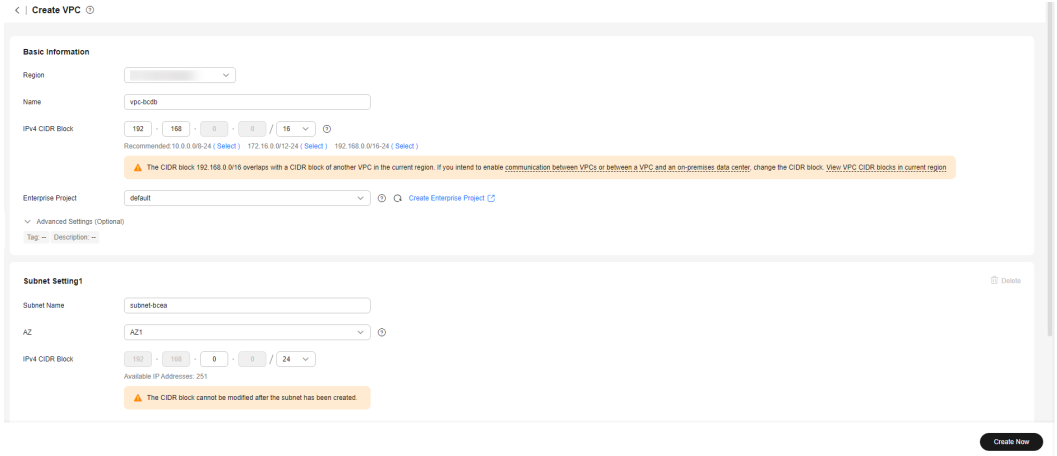
Create a VPC and security group for a DB instance in the production center.

Creating a VPC

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

Step 2 On that page, select **CN-Hong Kong** for **Region**, and configure the basic information, subnet, and IP address.

Figure 3-2 Creating a VPC



Step 3 Click **Create Now**.

----End

Creating a Security Group

Step 1 Log in to the **management console**.

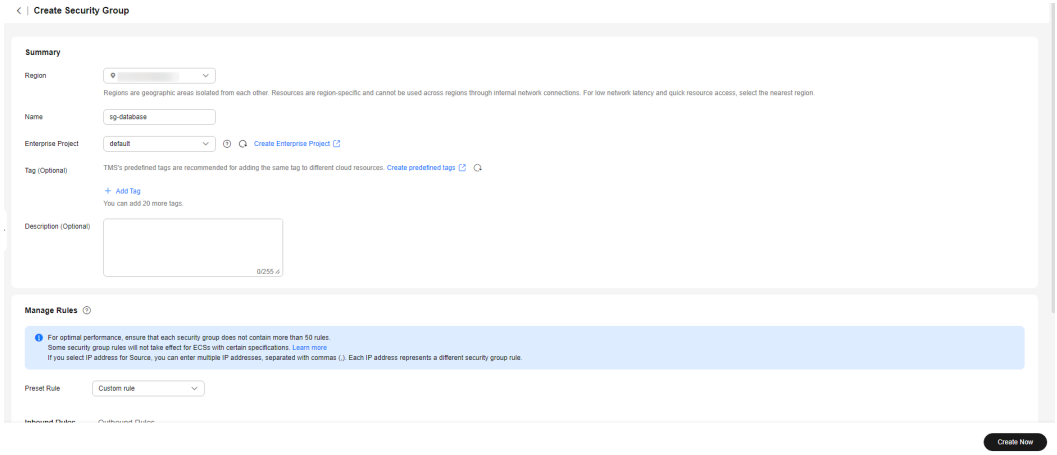
Step 2 Click  in the upper left corner and select **CN-Hong Kong**.

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 3-3 Creating a security group



Step 6 Click **Create Now**.

----End

3.4.2 Creating an EIP

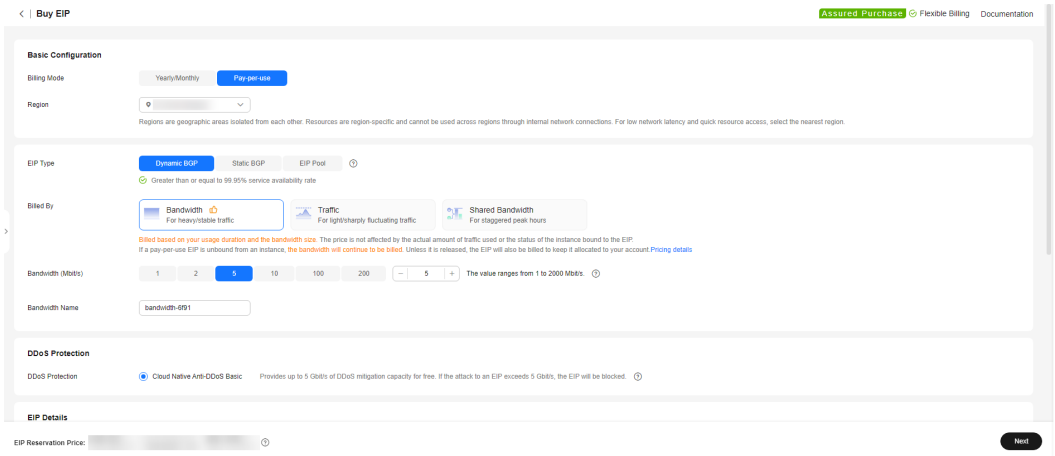
Create an EIP for your source DB instance. Using the EIP, external systems can access your application and DRS can connect to the source DB instance.

Procedure

Step 1 Go to the **Buy EIP** page.

Step 2 On that page, select **CN-Hong Kong** for **Region**, and configure the basic information and bandwidth as prompted.

Figure 3-4 Buying an EIP



Step 3 Click **Next**.

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

----End

3.4.3 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 3-5 Selecting a DB engine

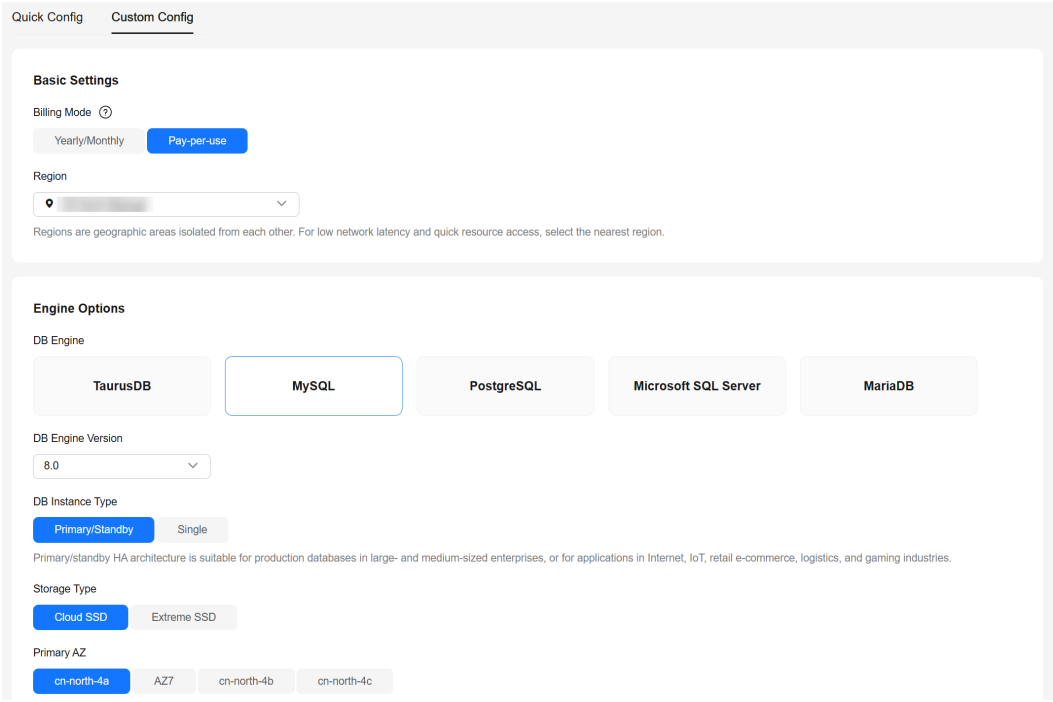


Figure 3-6 Selecting specifications

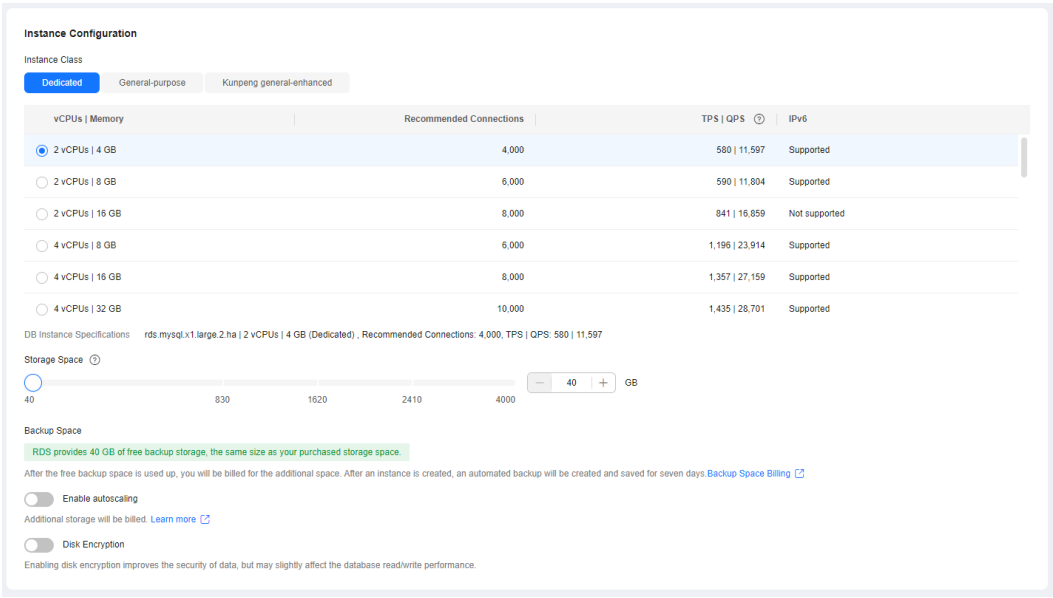


Figure 3-7 Configuring network information as planned

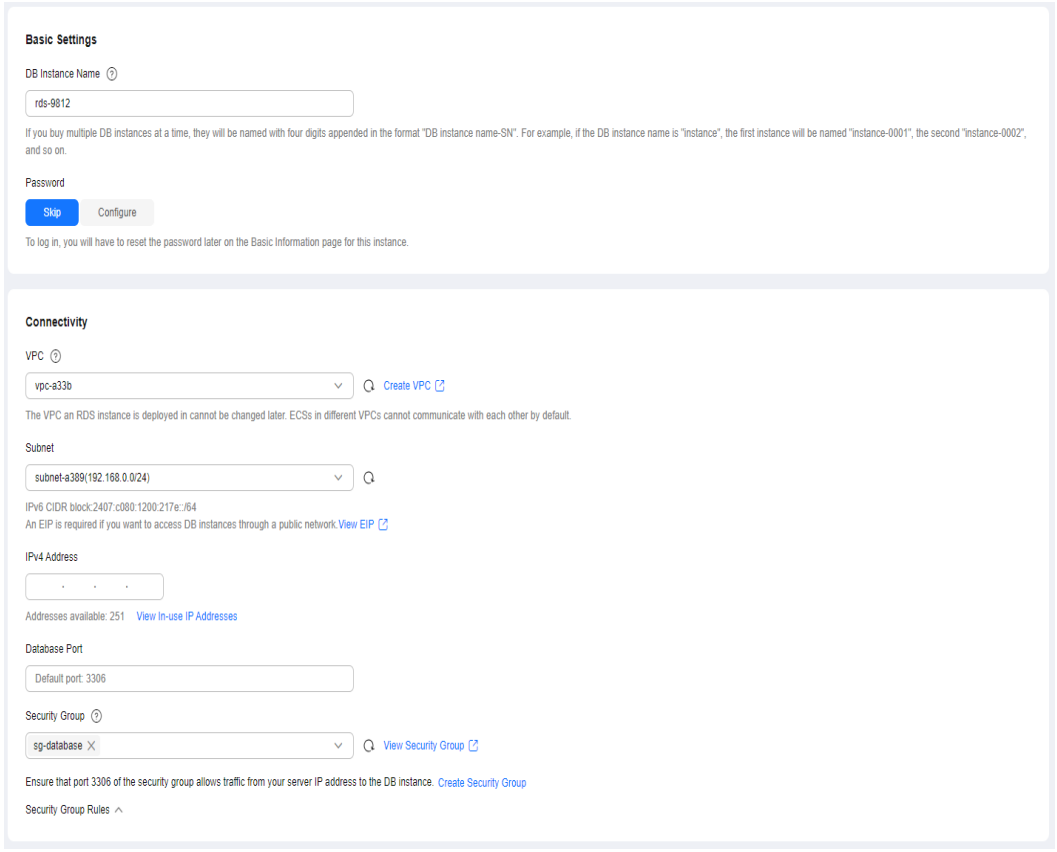
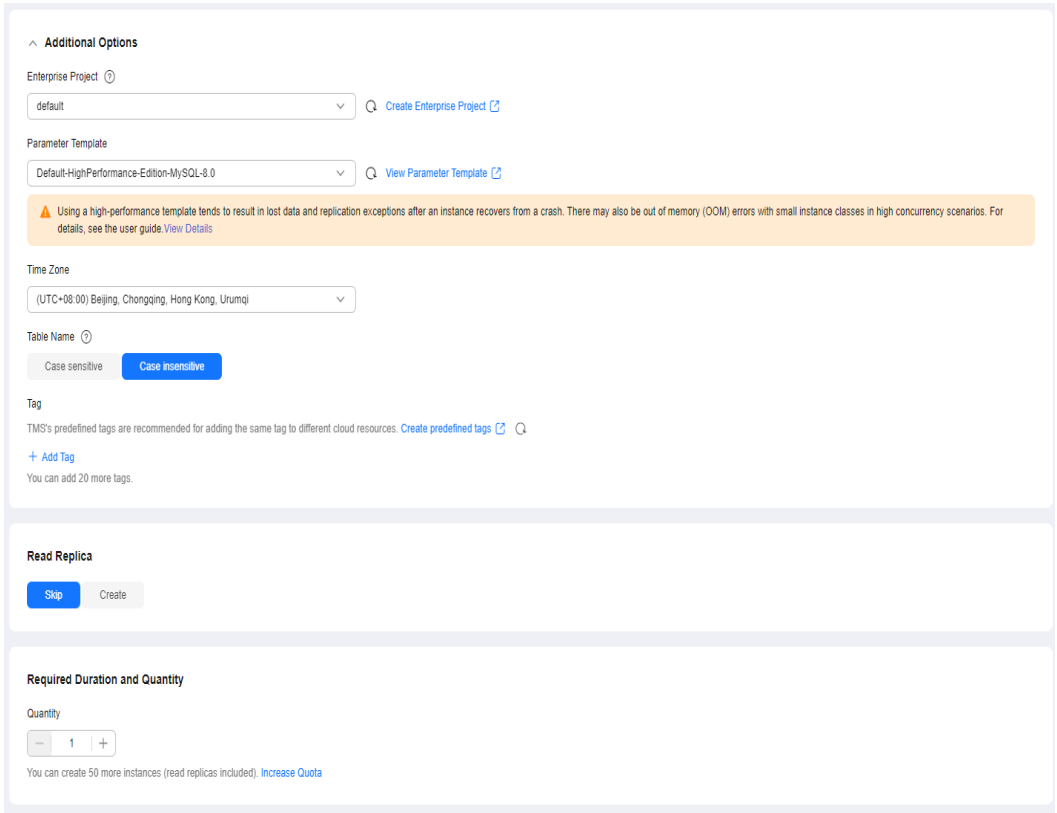


Figure 3-8 Additional options



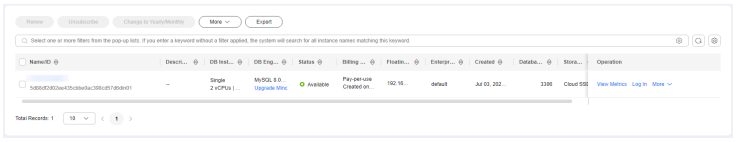
Step 3 Confirm the settings.

- 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.

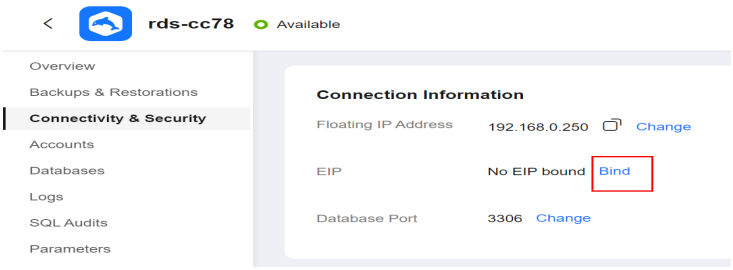
1. On the **Instances** page, click the instance name to go to the **Overview** page.

   Figure 3-9 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 **OK**.

   Figure 3-10 Binding an EIP

----End

3.5 Configuring an RDS for MySQL Instance in the DR Center

3.5.1 Creating a VPC and Security Group

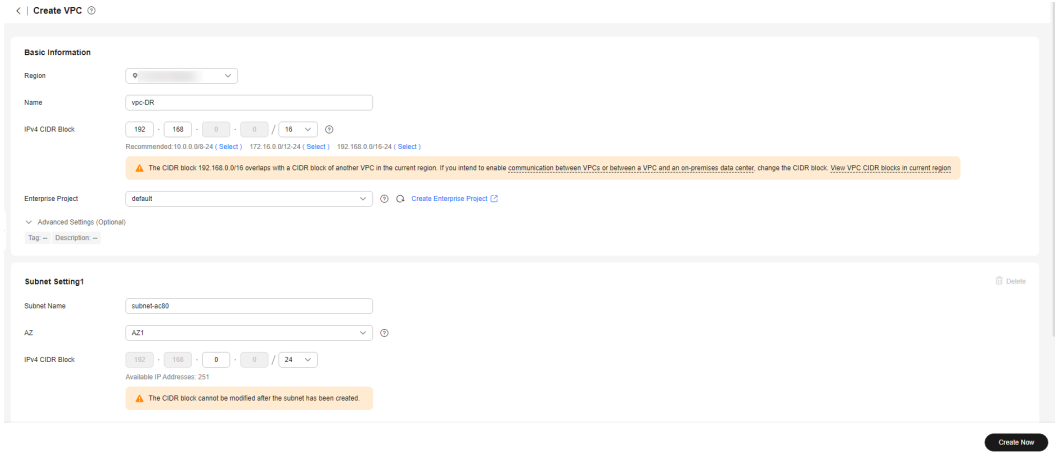
Create a VPC and security group for the DR instance to be configured, ensuring that it is in a different region from the instance created for production center.

Creating a VPC

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

Step 2 On that page, select **AP-Singapore** for **Region**, and configure the basic information, subnet, and IP address.

Figure 3-11 Creating a VPC



Step 3 Click **Create Now**.

----End

Creating a Security Group

Step 1 Log in to the **management console**.

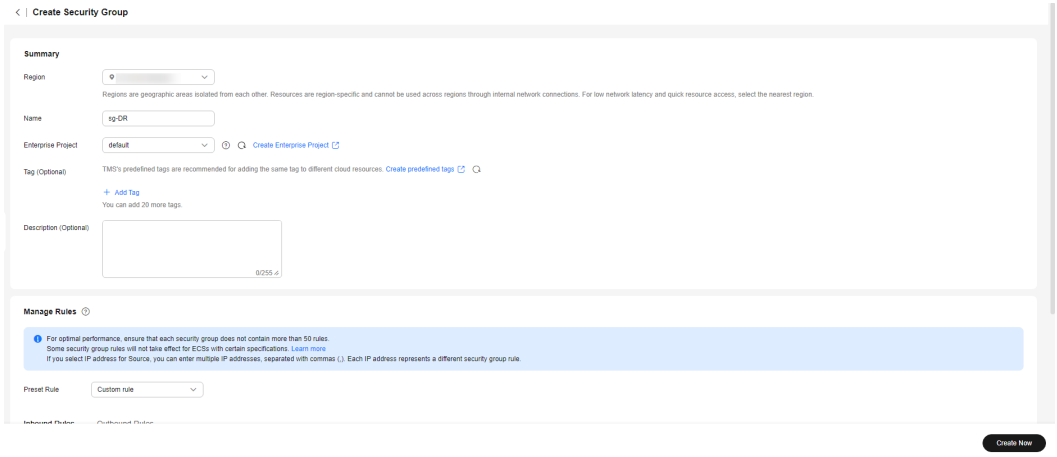
Step 2 Click  in the upper left corner of the management console and select **AP-Singapore**.

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 3-12 Creating a security group



Step 6 Click **Create Now**.

----End

3.5.2 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 3-13 Selecting a DB engine

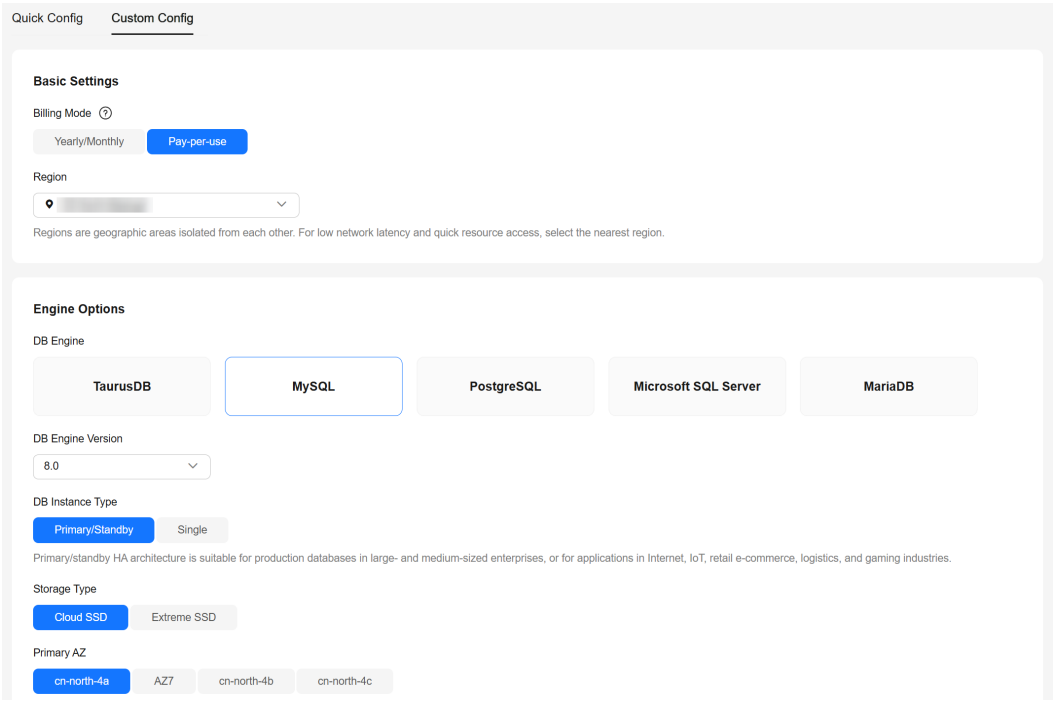


Figure 3-14 Selecting specifications

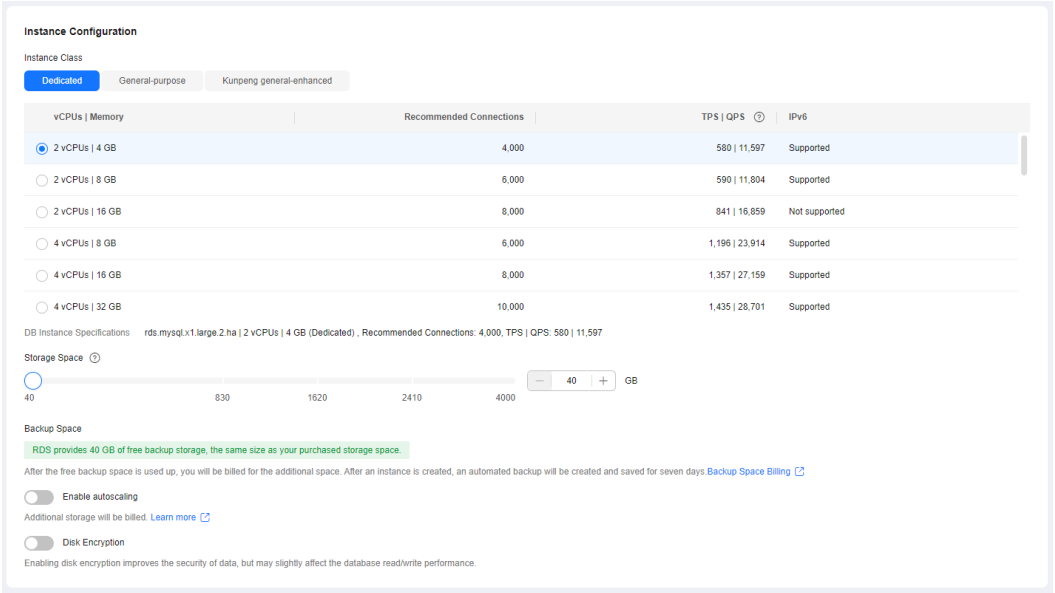


Figure 3-15 Configuring network information as planned

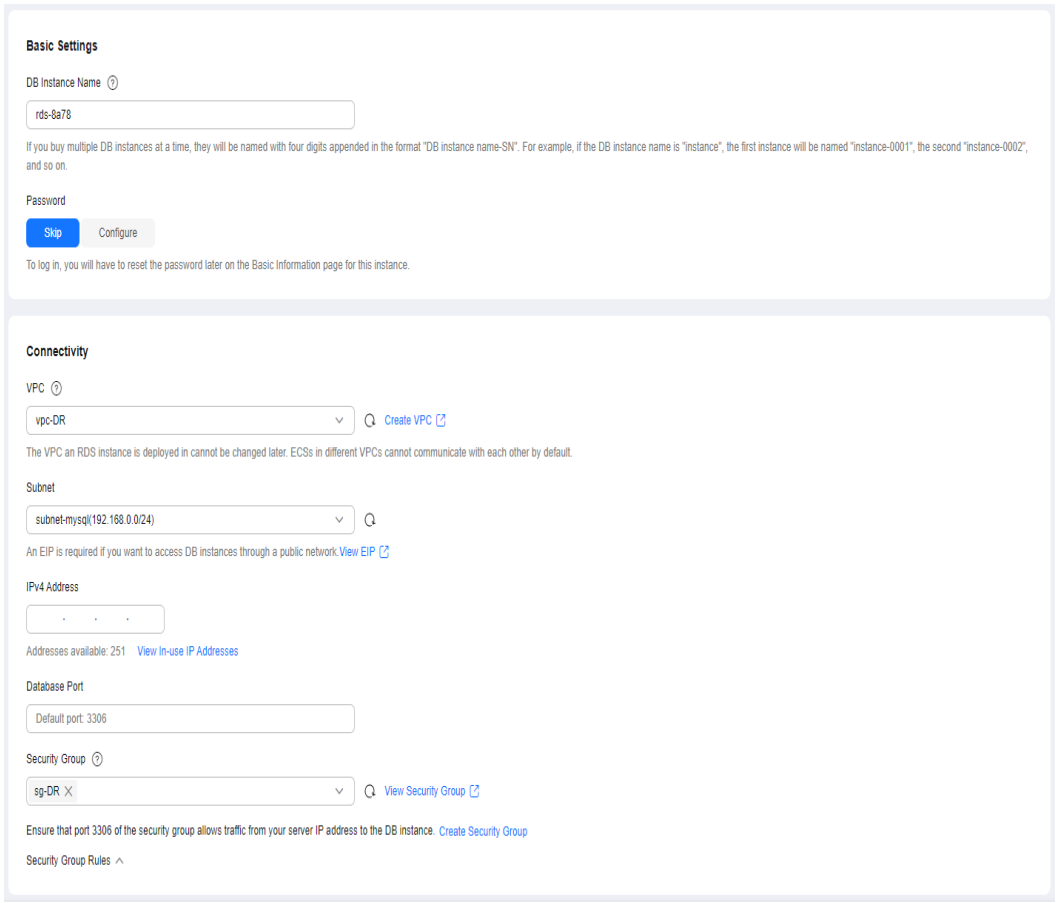
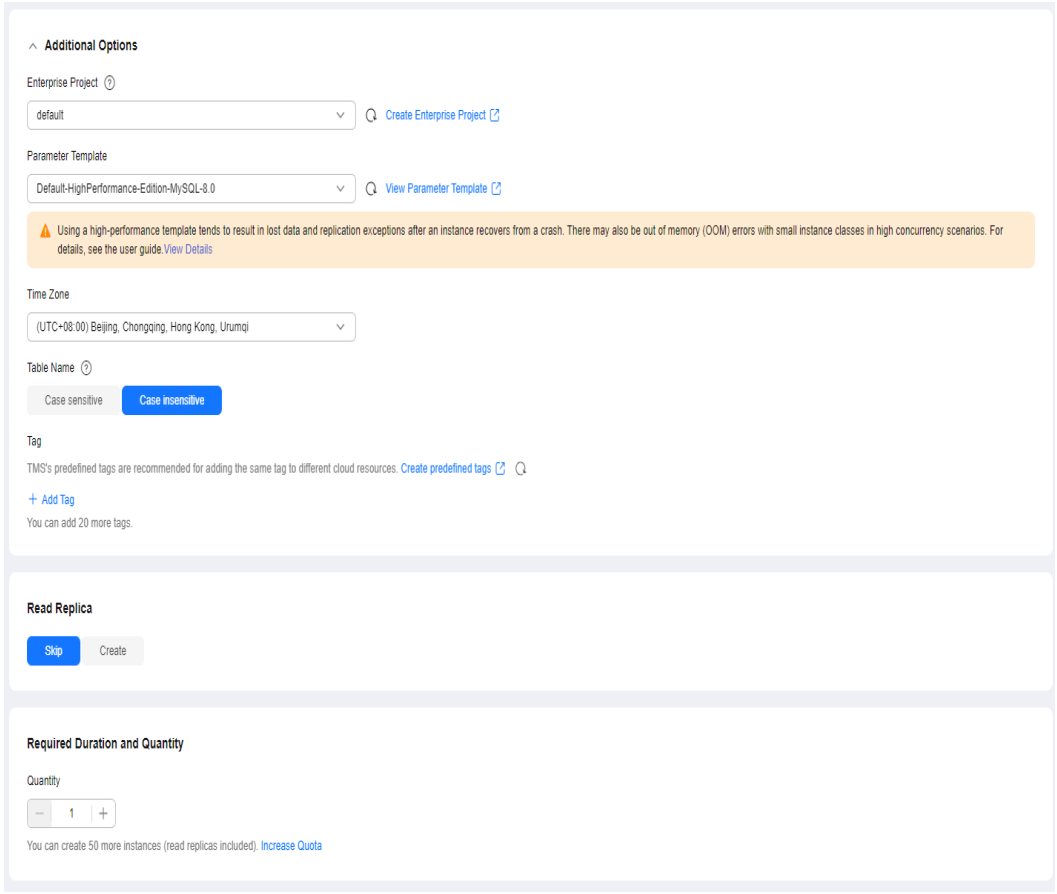


Figure 3-16 Additional options



Step 3 Confirm the settings.

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

----End

3.6 Configuring Remote Disaster Recovery

3.6.1 Creating a DRS Disaster Recovery Task

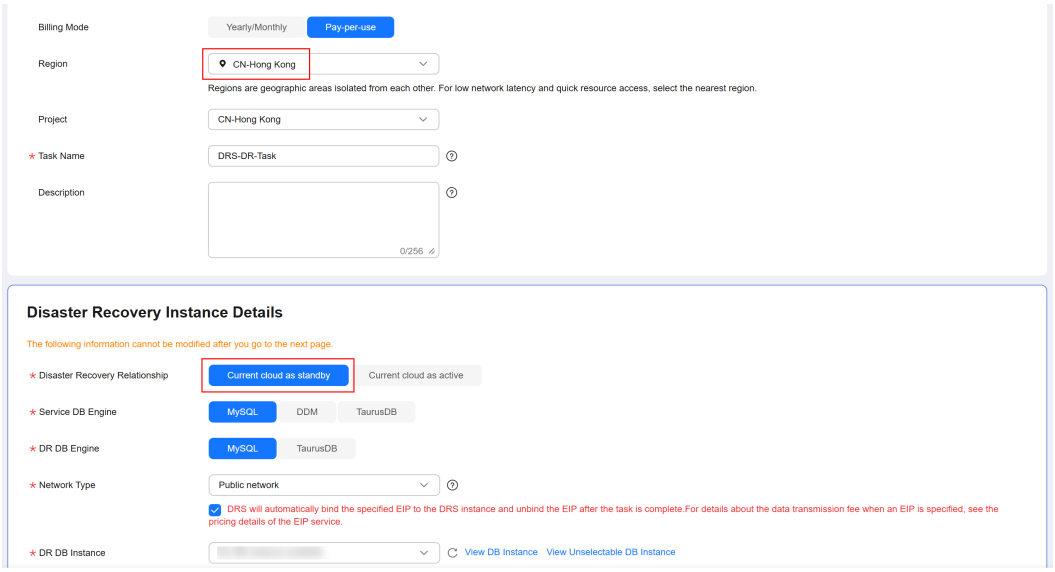
Create a DRS disaster recovery task in the same region as the RDS for MySQL instance configured for the DR center.

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 3-17 Setting DR instance information



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

----End

3.6.2 Configuring the Disaster Recovery Task

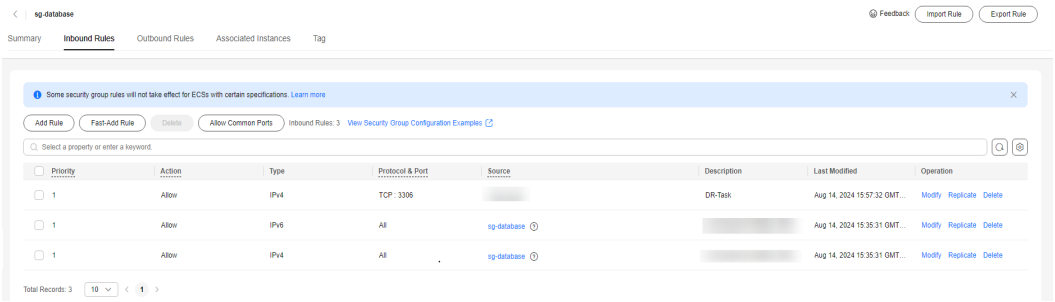
Configure the disaster recovery task, including setting the source and destination databases.

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 3-18 Adding a security group rule

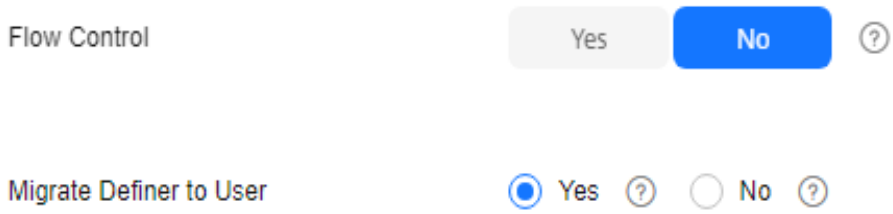


In the **Source Database** area, set **IP Address or Domain Name** and **Port** to the EIP and port of the RDS for MySQL instance in the production center. When the connection test is successful, click **Next**.

Figure 3-19 Editing a disaster recovery task

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

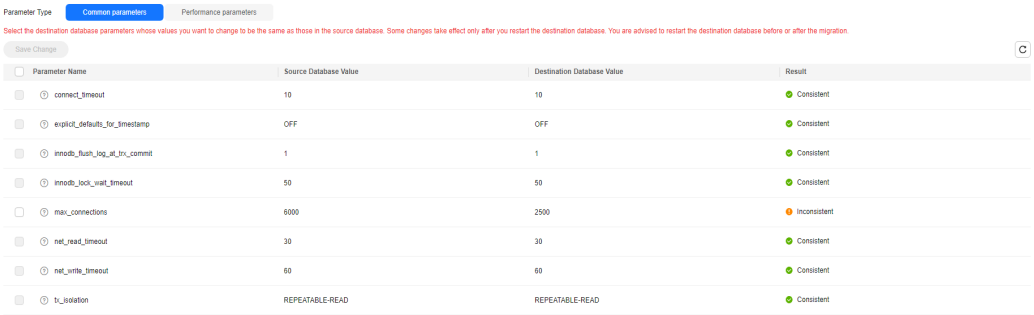
Figure 3-20 Configuring flow control



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

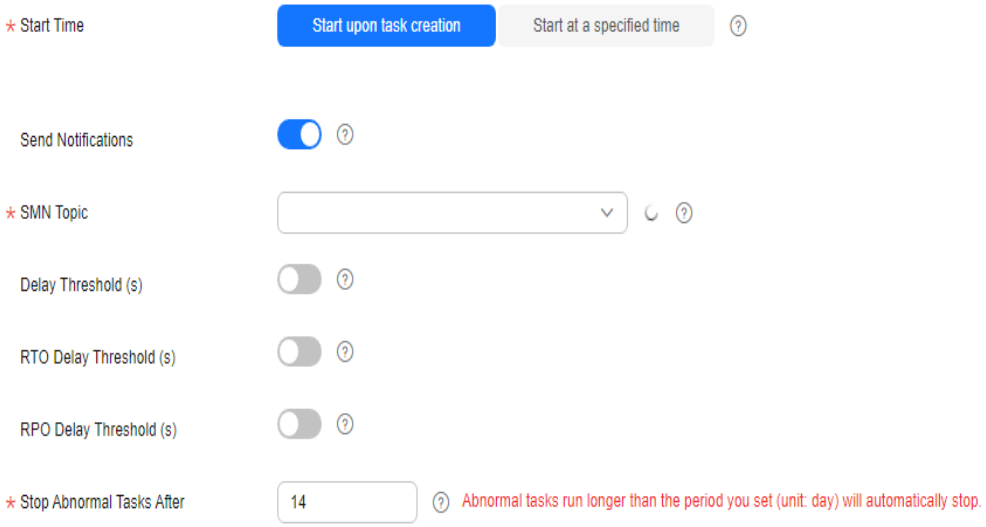
Step 5 Configure parameters and click **Next**.

Figure 3-21 Configuring parameters



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

Figure 3-22 Starting the task



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

3.6.3 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.

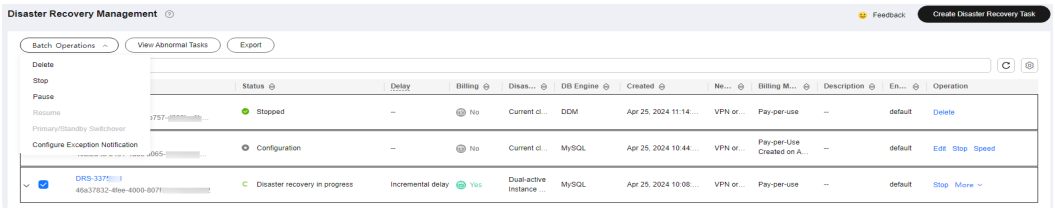
Step 2 Receive an SMN email notification.

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 3-23 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

4 Migrating MySQL Databases from Other Clouds to RDS for MySQL

4.1 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 greater than or equal to $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**.

4.2 Resource Planning

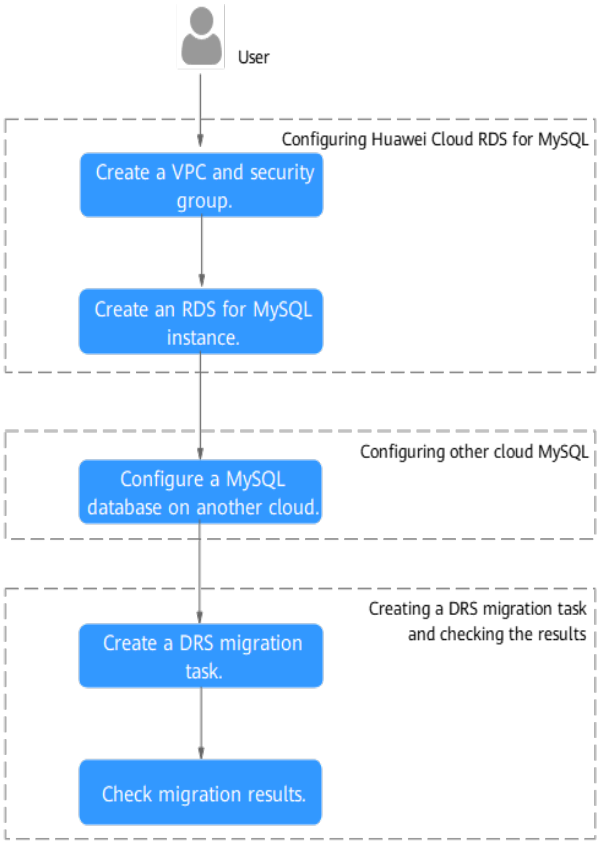
Table 4-1 Resource planning

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	-

Category	Subcategory	Planned Value	Description
DRS migration task	Task name	DRS-mysql	Specify a name that is easy to identify.
	Source DB engine	MySQL	-
	Destination DB engine	MySQL	-
	Network type	Public network	Public network is used in this example.

4.3 Operation Process

Figure 4-1 Flowchart



4.4 Creating a VPC and Security Group

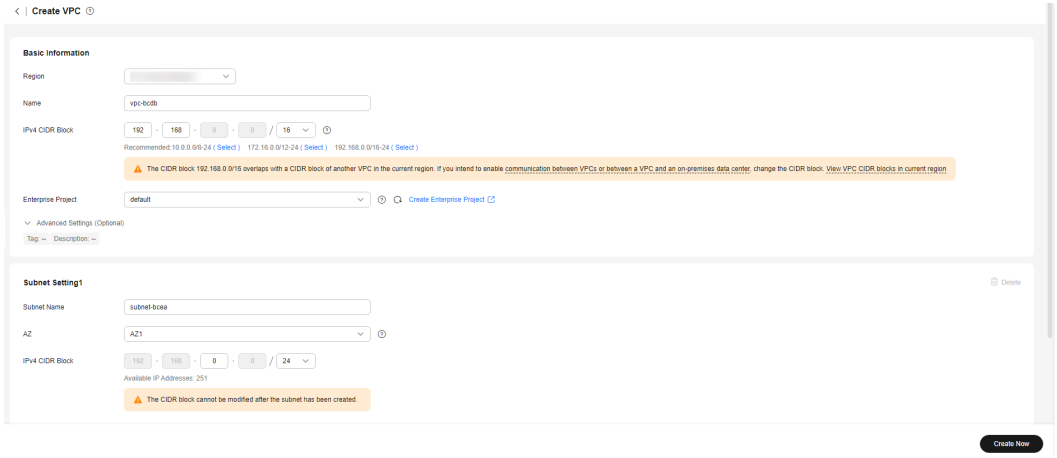
Create a VPC and security group for an RDS for MySQL instance

Creating a VPC

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

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

Figure 4-2 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**.

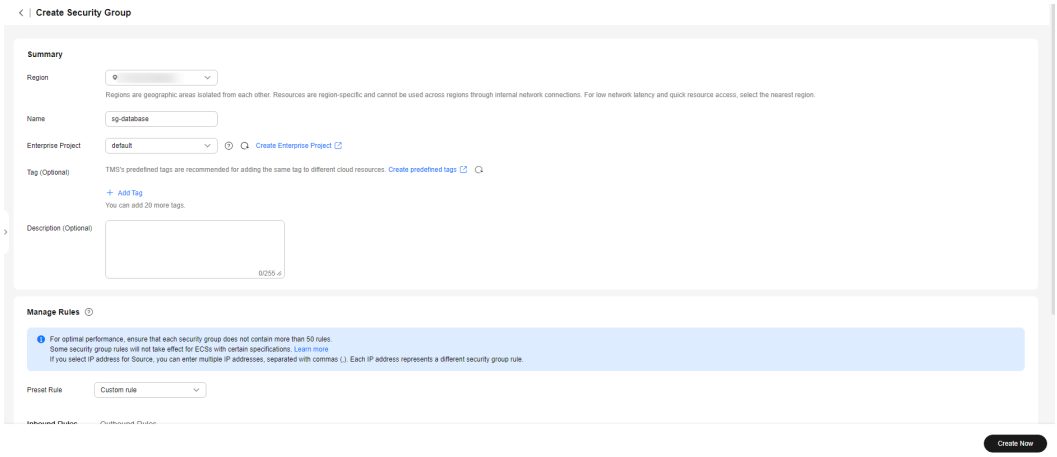
Step 3 Under the service list, choose **Networking** > **Virtual Private Cloud**.

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

Step 5 Click **Create Security Group**.

Step 6 Configure parameters as needed.

Figure 4-3 Creating a security group



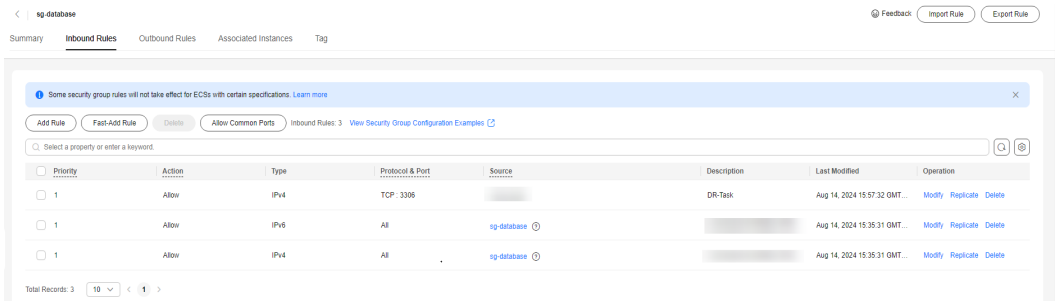
Step 7 Click **Create Now**.

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 4-4 Inbound rules



----End

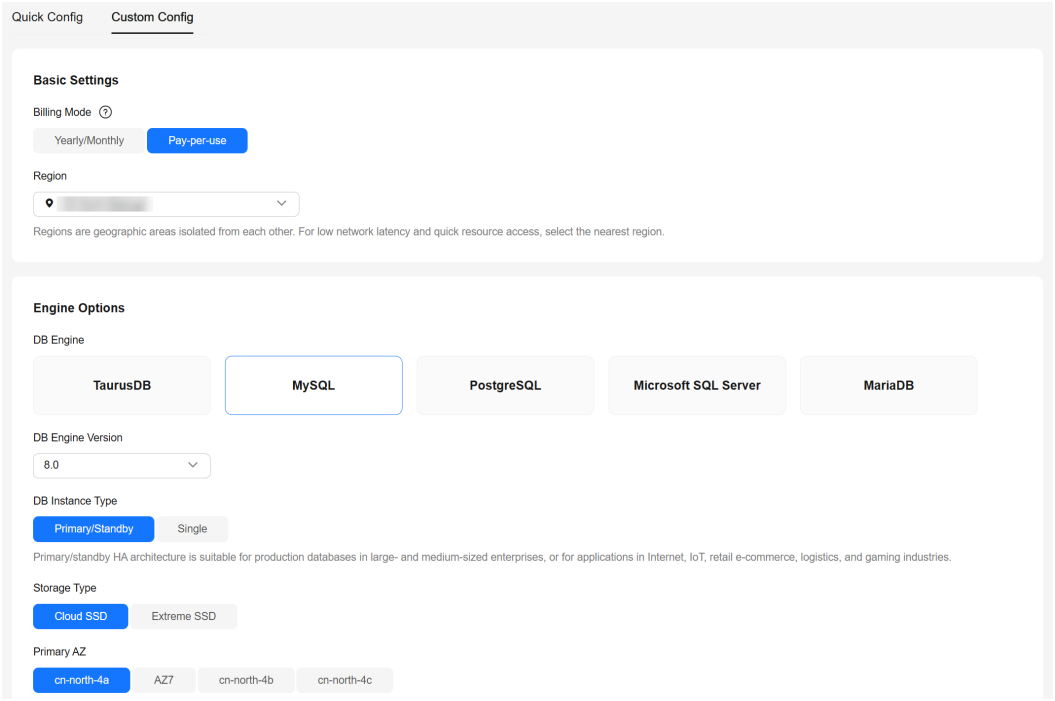
4.5 Creating an RDS for MySQL Instance

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.

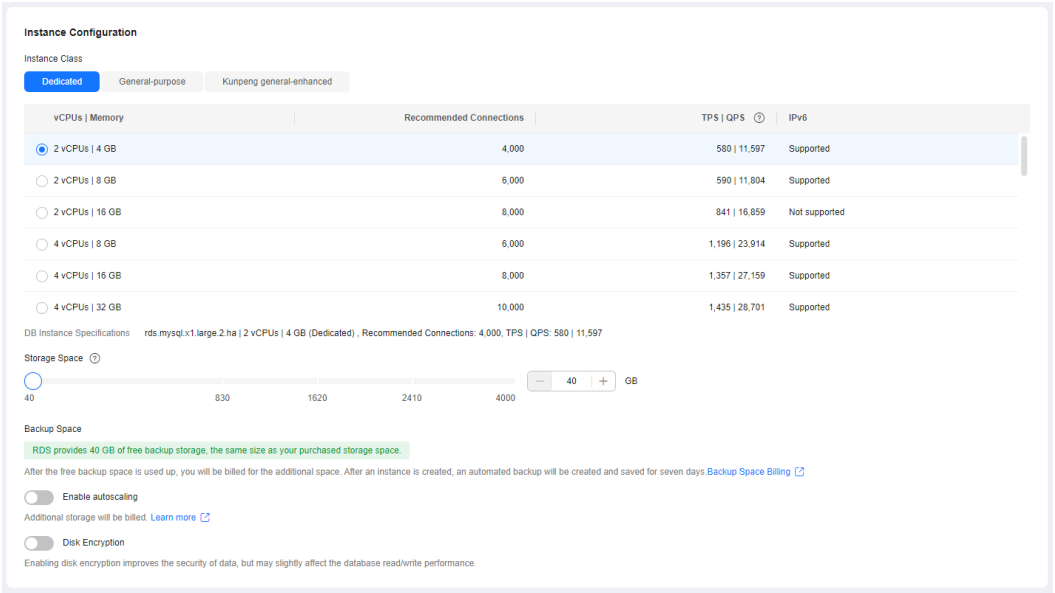
Step 2 Configure basic information for the instance. Select **CN-Hong Kong** for **Region**.

Figure 4-5 Basic information



Step 3 Select an instance class.

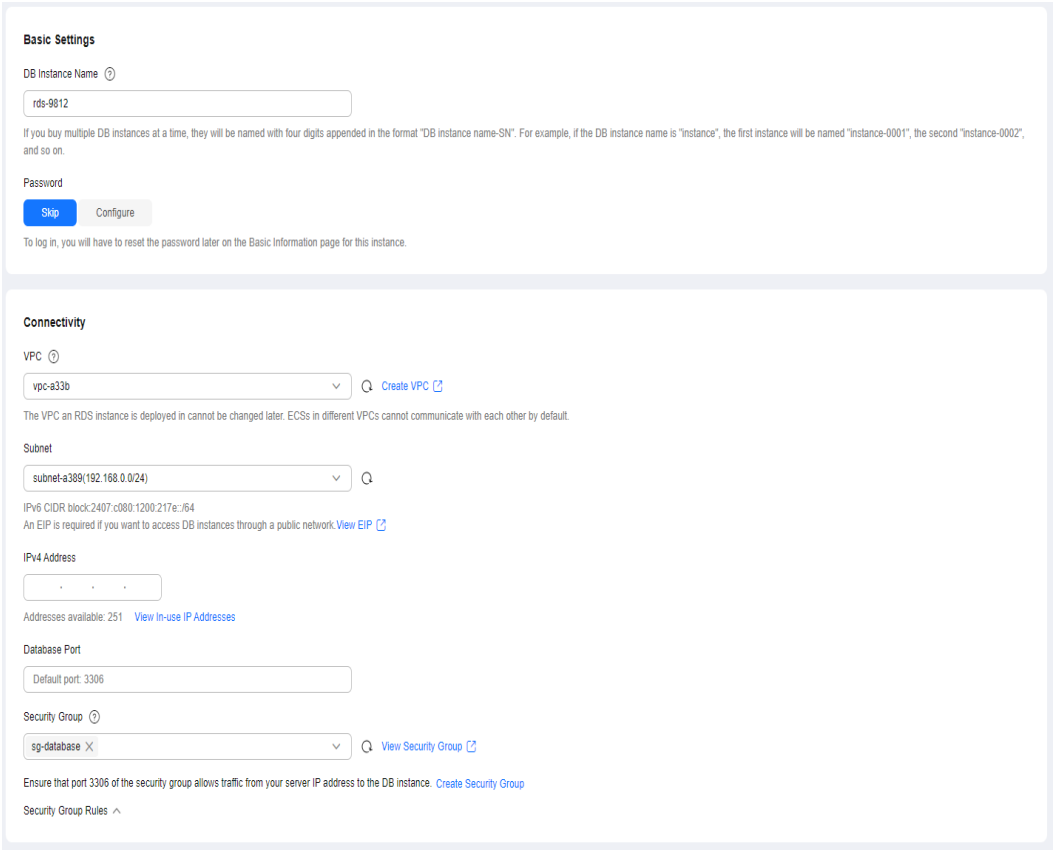
Figure 4-6 Instance class



Step 4 Select a VPC and security group for the instance and configure the database port.

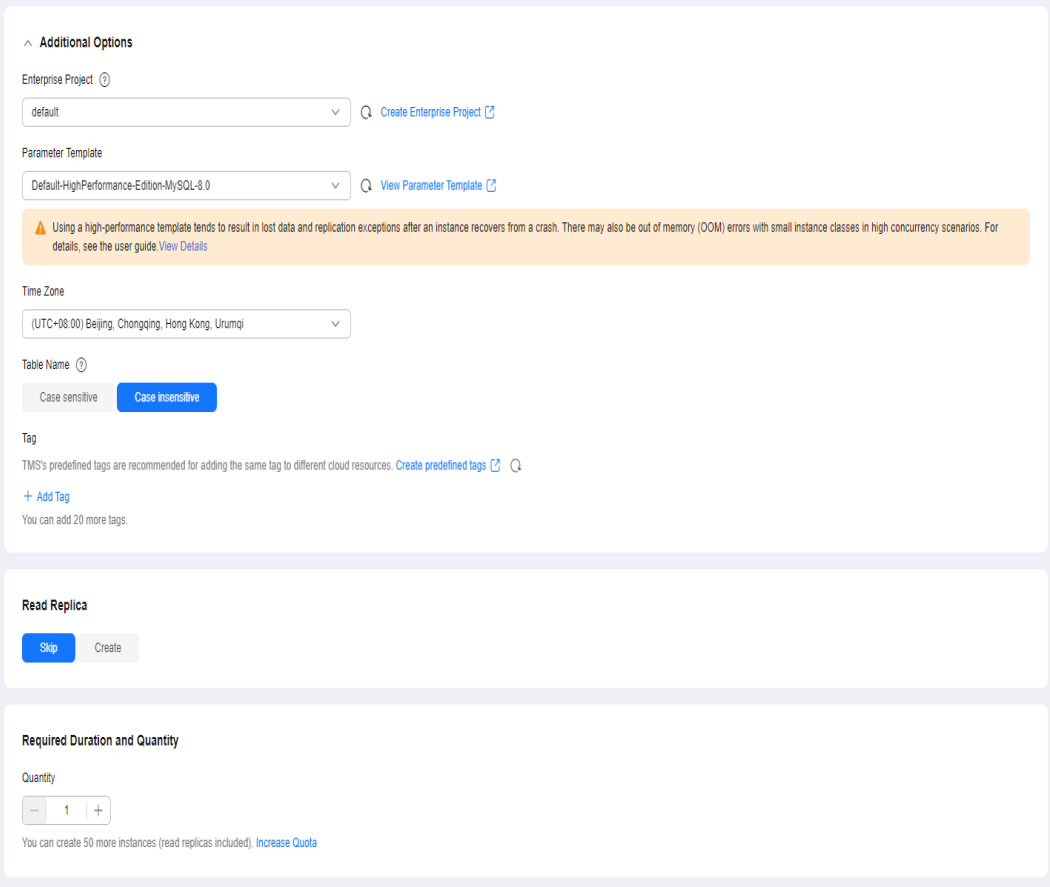
The VPC and security group have been created in **Creating a VPC and Security Group**.

Figure 4-7 Network configurations



Step 5 Complete advanced settings.

Figure 4-8 Additional options



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

4.6 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 4-2 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 4-2 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.

4.7 Cloud Migration

4.7.1 Creating a DRS Migration Task

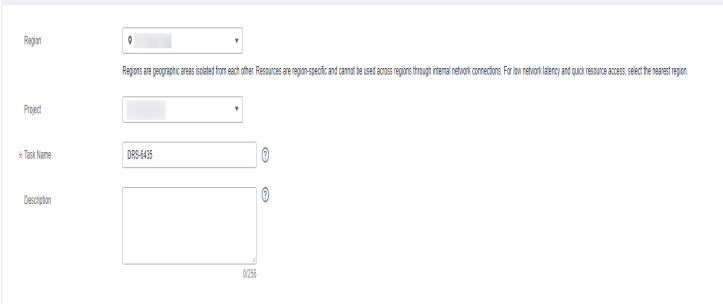
Creating a Migration Task

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**.

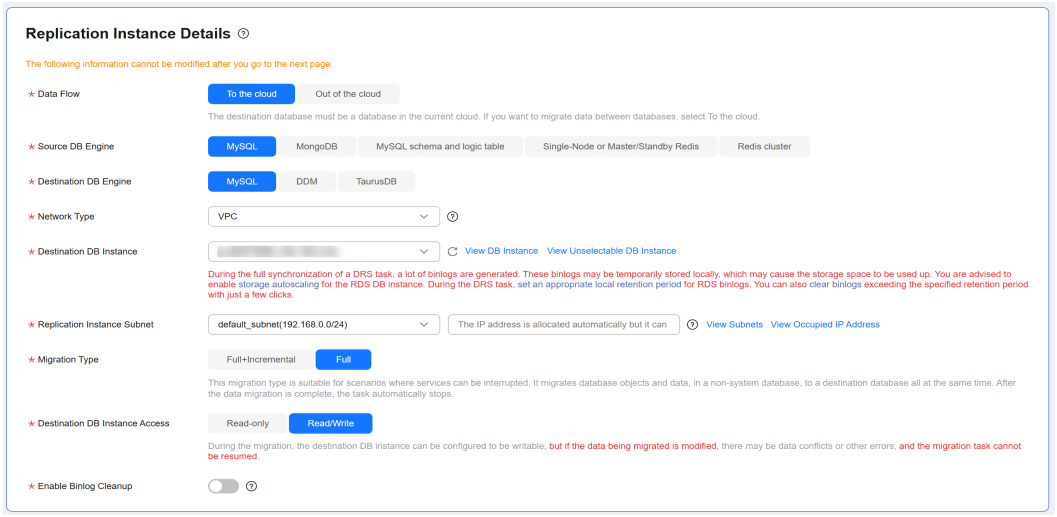
Figure 4-9 Migration task



2. Configure the replication instance information.

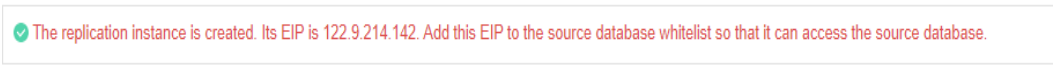
   Select the RDS instance created in **Creating an RDS for MySQL Instance** as the destination database.

   Figure 4-10 Replication instance details



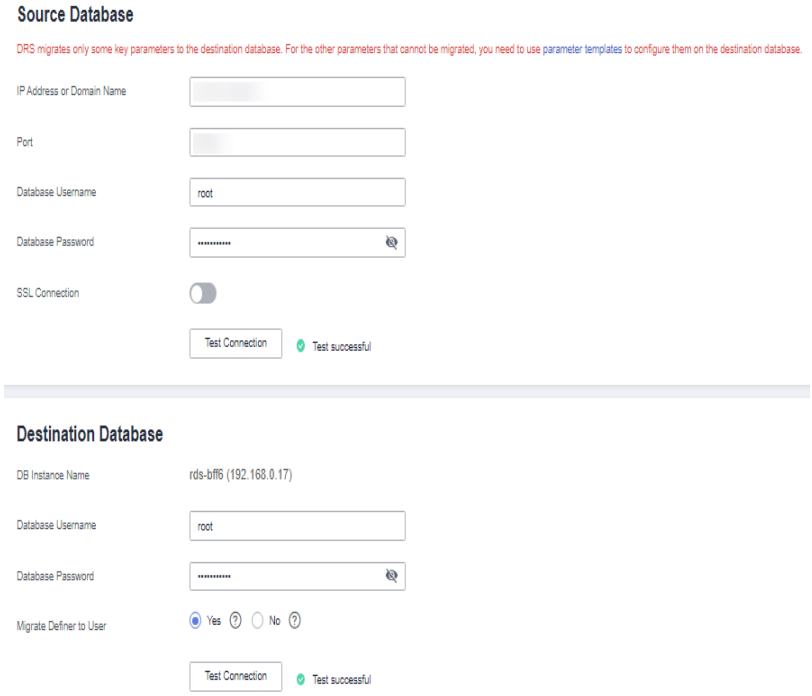
Step 3 Click **Create Now**.

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



Step 4 Configure the source and destination database information.

Figure 4-11 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

4.7.2 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.

5 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.

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.

Procedure

Step 1: Configure the Network

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

Step 2: Buy an ECS

Step 1 Log in to the **management console**.

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

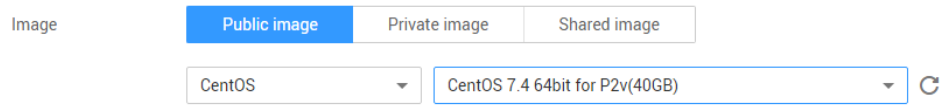
Step 3 Choose **Compute** > **Elastic Cloud Server**. The **Elastic Cloud Server** page is displayed.

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 5-1**.

   Figure 5-1 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

Step 3: Set Up Your LAMP Environment

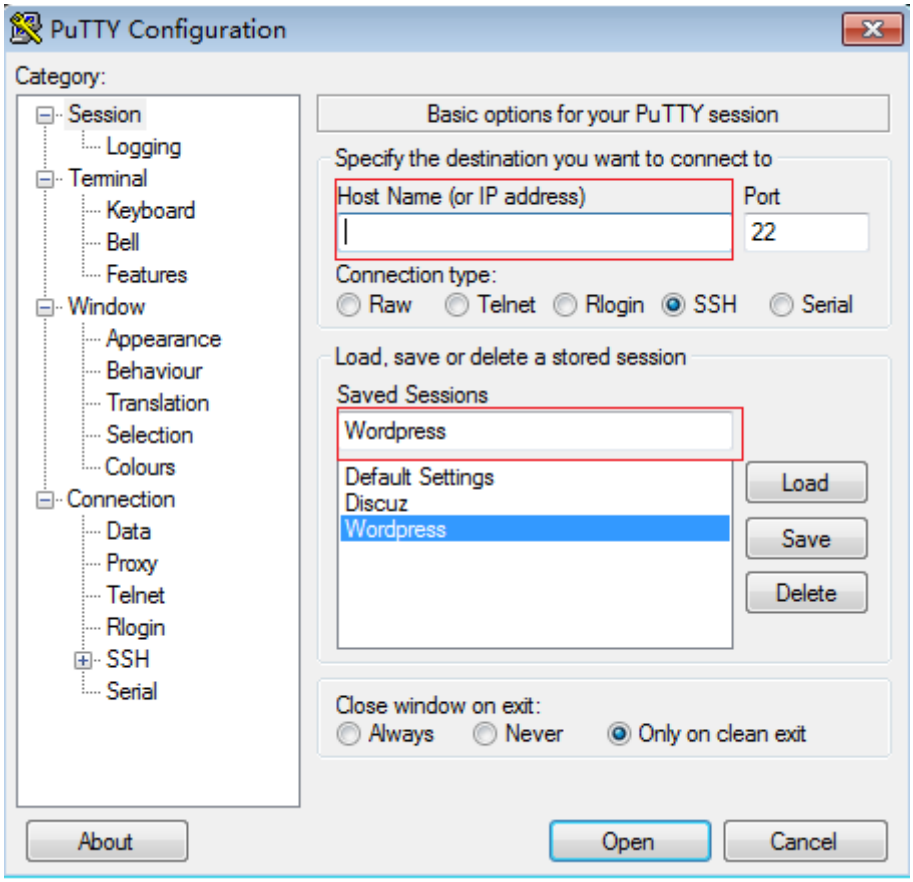
Step 1 Download the PuTTY client.

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 5-2**.

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 5-2 Configuring PuTTY



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

Step 4: Buy and Configure an RDS DB Instance

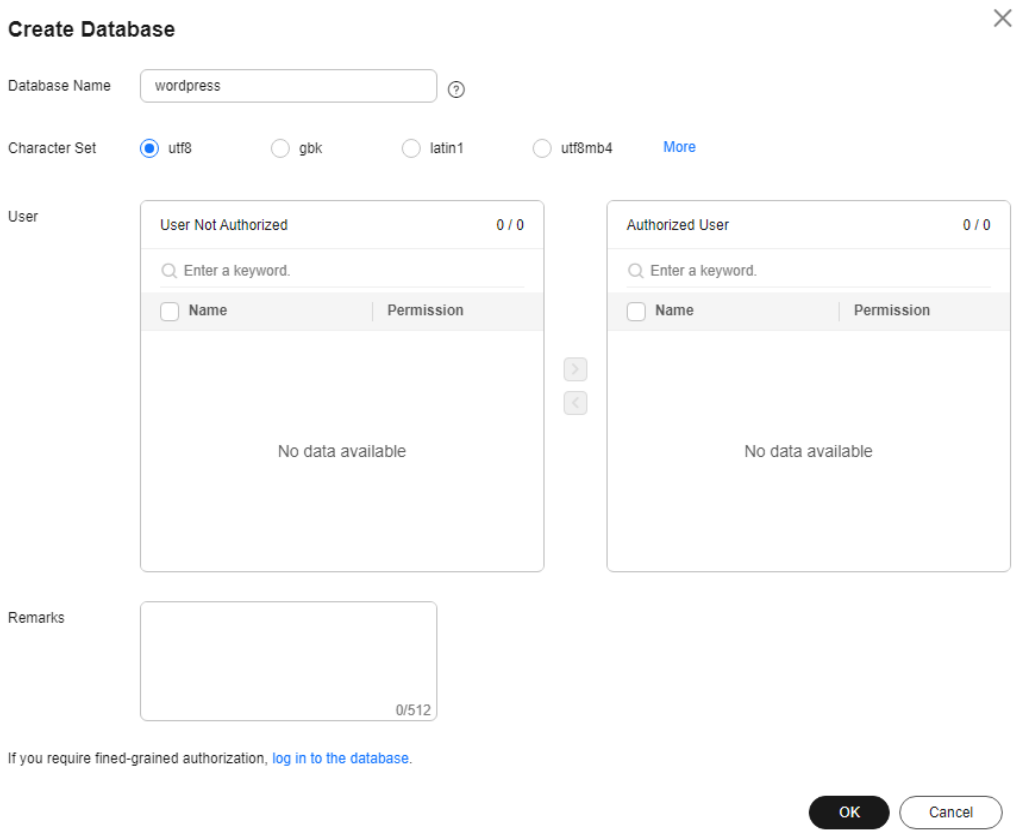
Step 1 **Buy an RDS for MySQL DB instance** as required.

- 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.

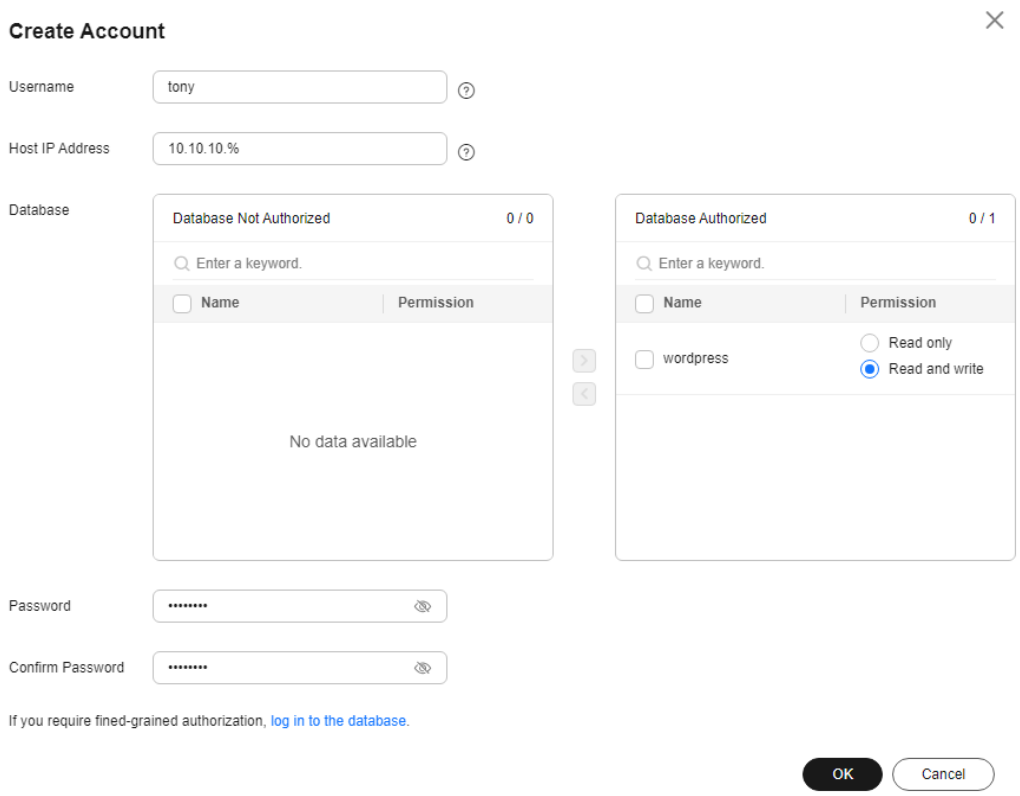
Step 3 Choose **Databases** in the navigation pane on the left and click **Create Database**. In the displayed dialog box, enter a database name, such as ***wordpress***, select a character set, and authorize permissions for database users. Then, click **OK**.

Figure 5-3 Creating a 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 5-4 Creating an account



----End

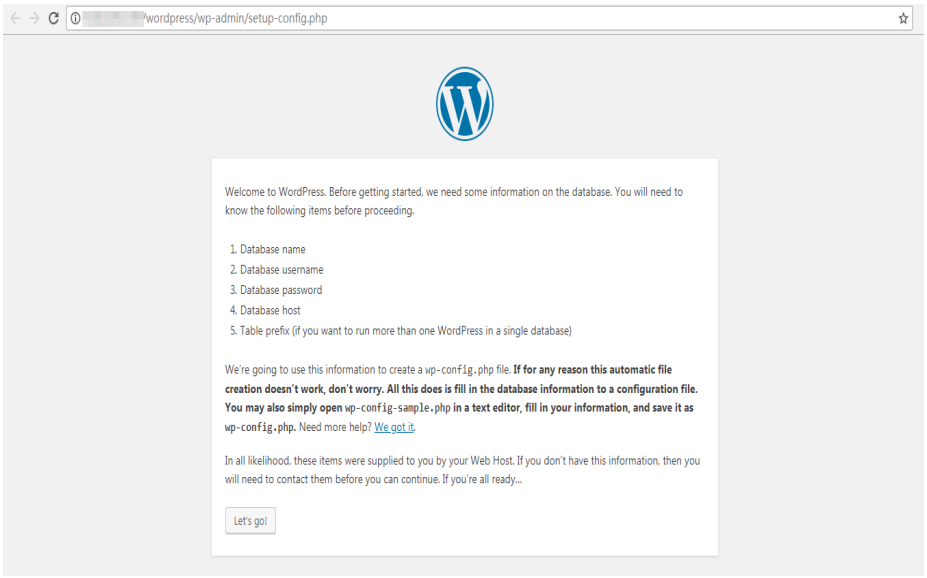
Step 5: Install WordPress

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 **Step 2: Buy an ECS**.

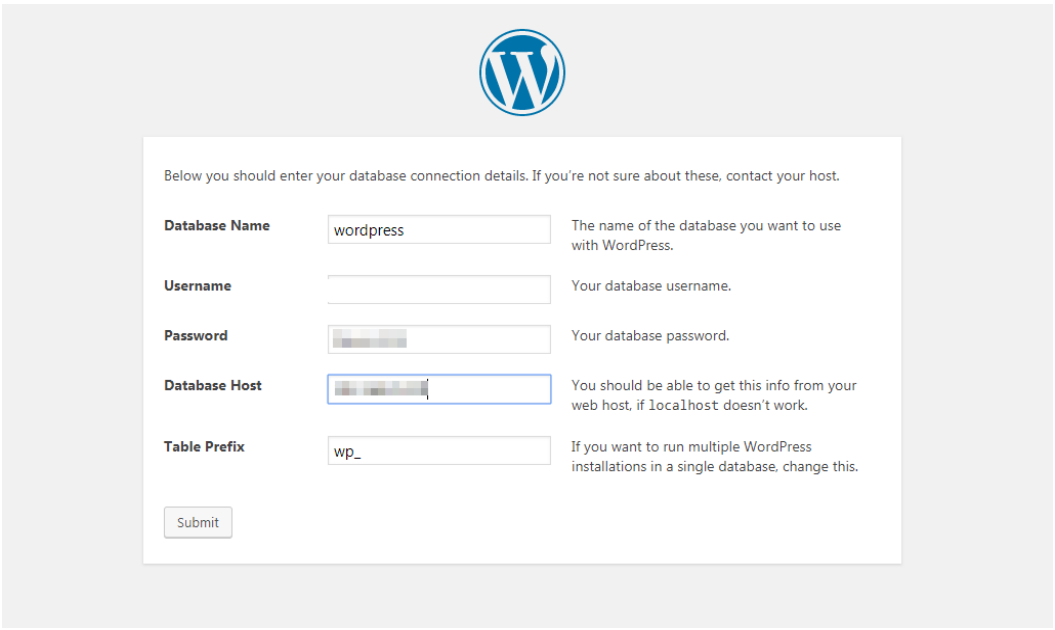
Figure 5-5 Visiting WordPress



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

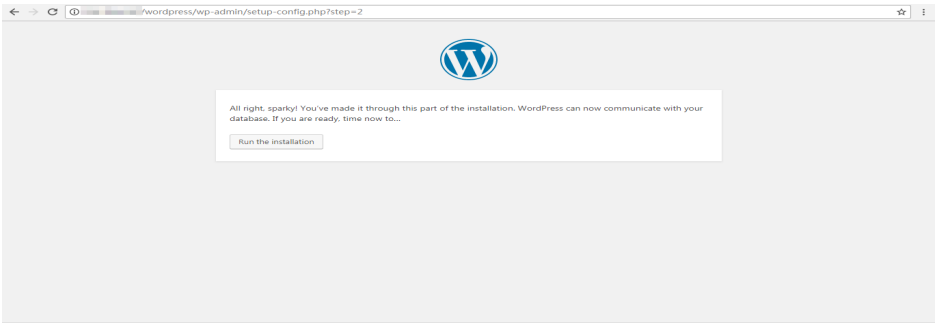
- The database name is ***wordpress***.
- The username is ***tony***.
- The password is the one that you set for ***tony***.
- The database host is the floating IP address of DB instance rds-01.

Figure 5-6 Entering database connection information



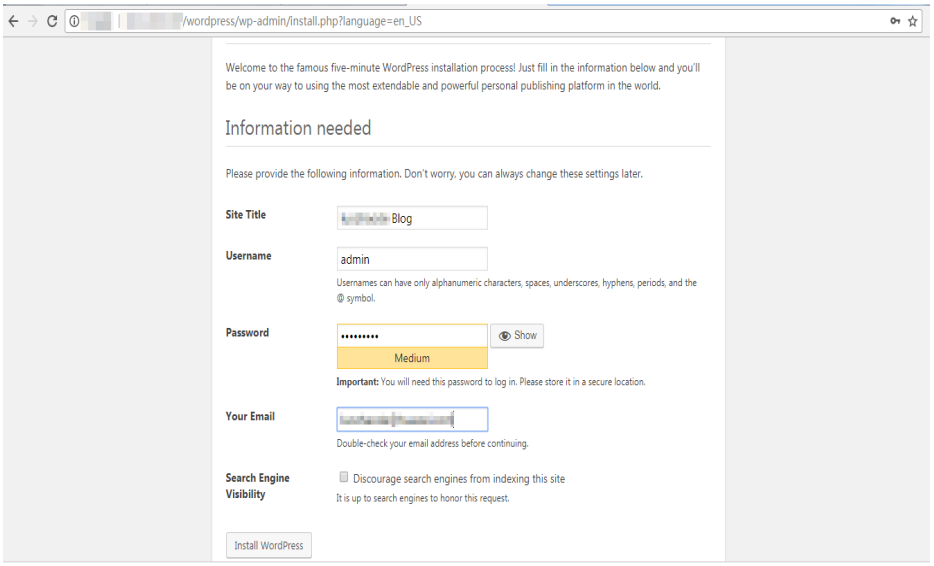
Step 4 After the database connection details are verified, click **Run the installation**.

Figure 5-7 Running the installation



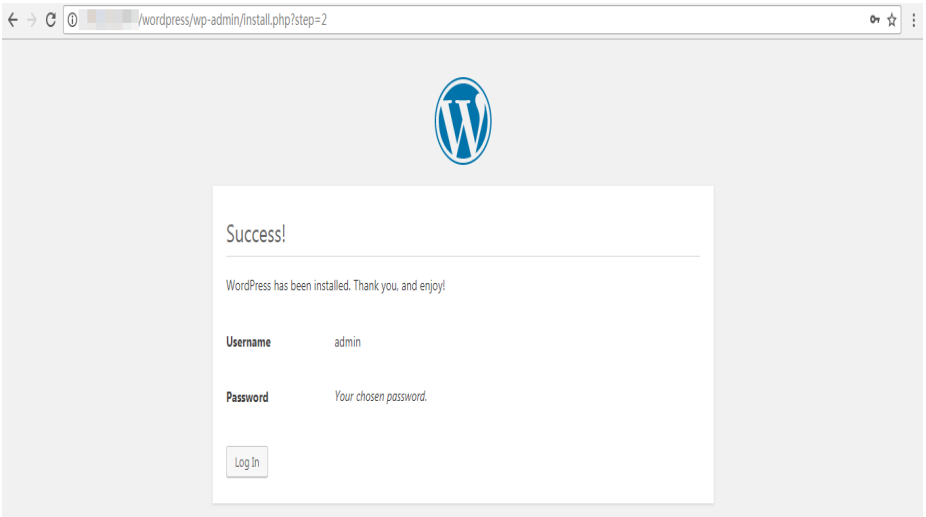
Step 5 Set **Site Title**, **Username**, and **Password** for logging in to your blog. Then, click **Install WordPress**.

Figure 5-8 Setting basic information



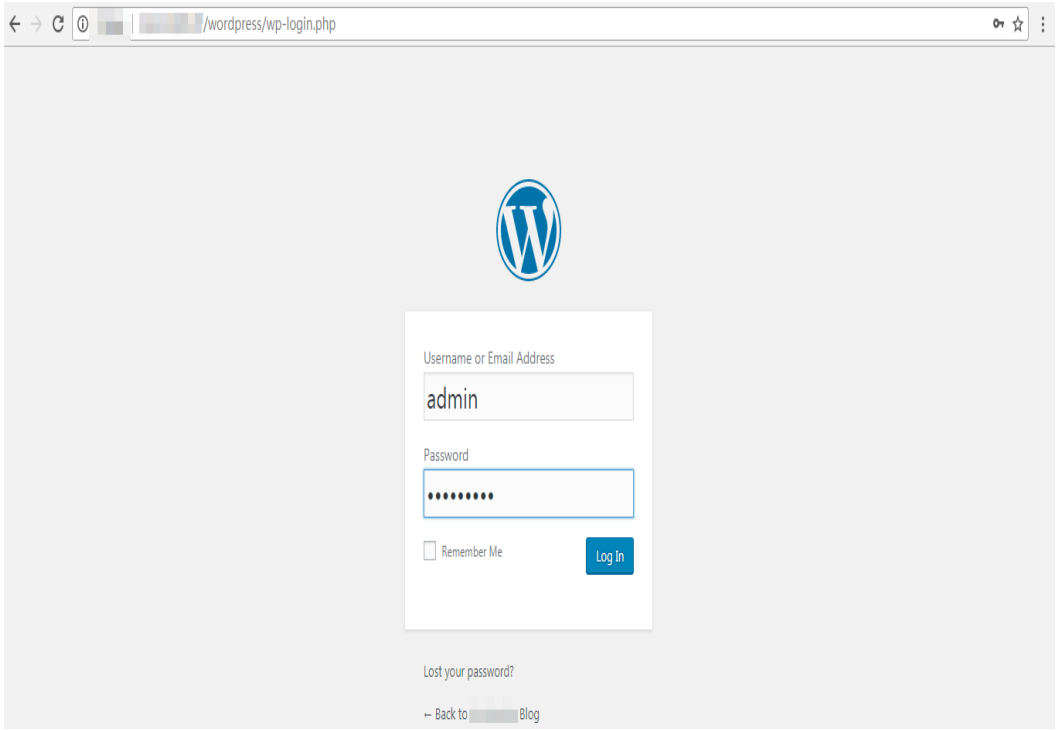
Step 6 Click **Log In** after WordPress has been successfully installed.

Figure 5-9 Successful installation



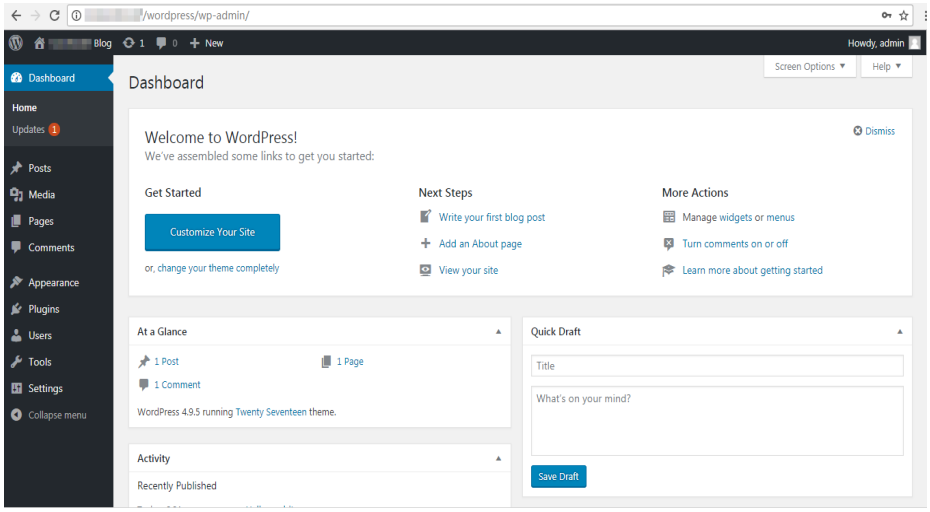
Step 7 Enter the username and password on the displayed login page. Then, click **Log In**.

Figure 5-10 Logging in



Step 8 Check that WordPress has been deployed successfully.

Figure 5-11 Verification



----End

6 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. **Step 1: Configure the Network**
2. **Creating an ECS**
3. **Step 3: Set Up Your LAMP Environment**
4. **Step 4: Buy and Configure an RDS DB Instance**
5. **Step 5: Install 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.

Procedure

Step 1: Configure the Network

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

Step 2: Buy an ECS

Step 1 Log in to the **management console**.

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

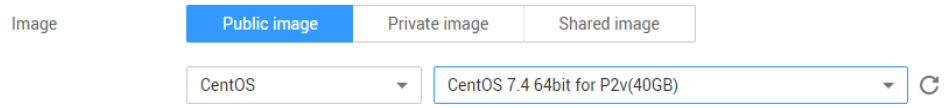
Step 3 Choose **Compute** > **Elastic Cloud Server**. The **Elastic Cloud Server** page is displayed.

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 6-1**.

   Figure 6-1 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

Step 3: Set Up Your LAMP Environment

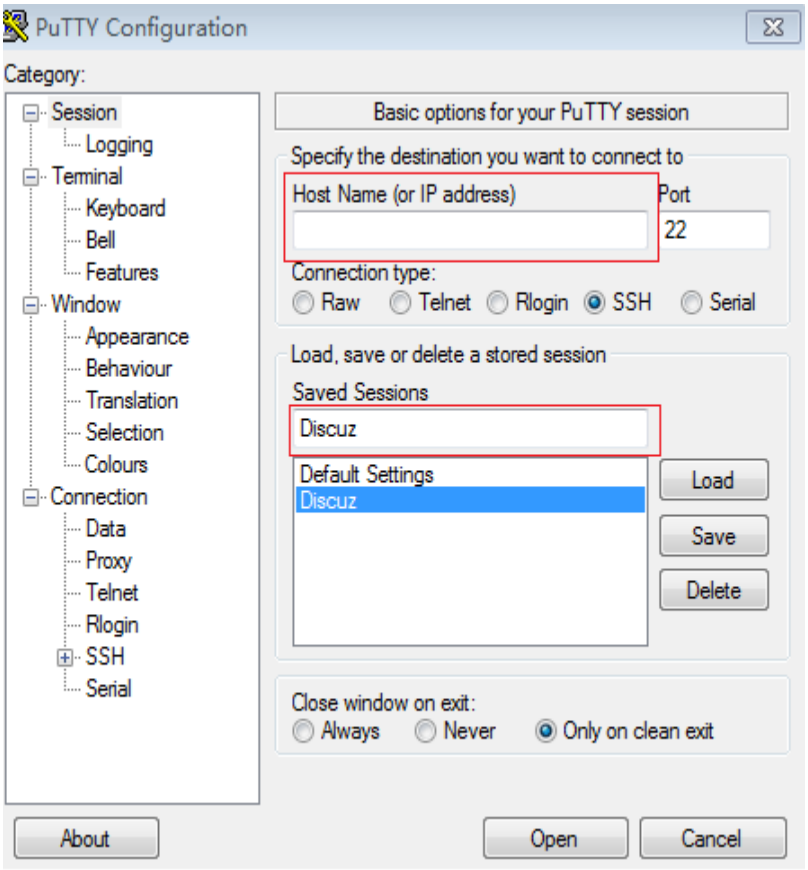
Step 1 Download the PuTTY client.

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 6-2**.

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 6-2 Configuring PuTTY



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

Step 4: Buy and Configure an RDS DB Instance

Step 1 **Buy an RDS for MySQL 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

Step 5: Install 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 **Step 2: Buy 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

7 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 7-1 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 7-2 Recommended configurations

innodb_flush_log_at_trx_commit	sync_binlog	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.

8 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 **Introduction to 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 join queries.
- Specify a field or add a WHERE clause, which will prevent full table scanning triggered by the SELECT statement.

9 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 9-1 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.

Parameter	Description
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*

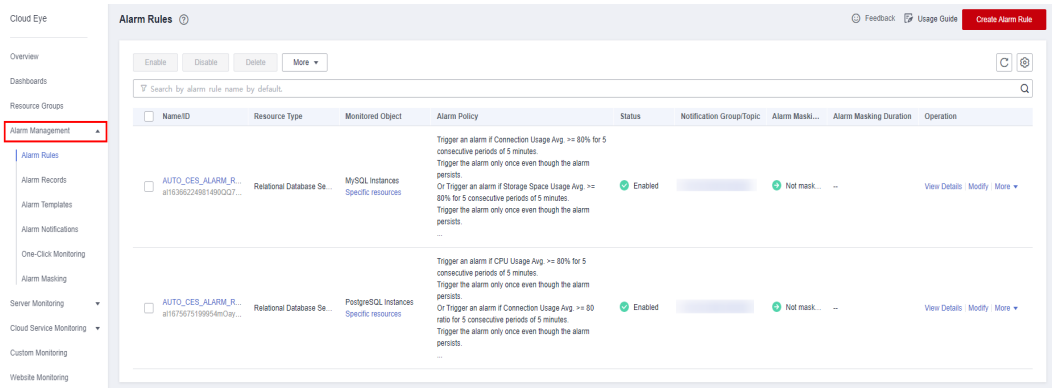
NOTICE

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 9-1 Viewing alarm rules



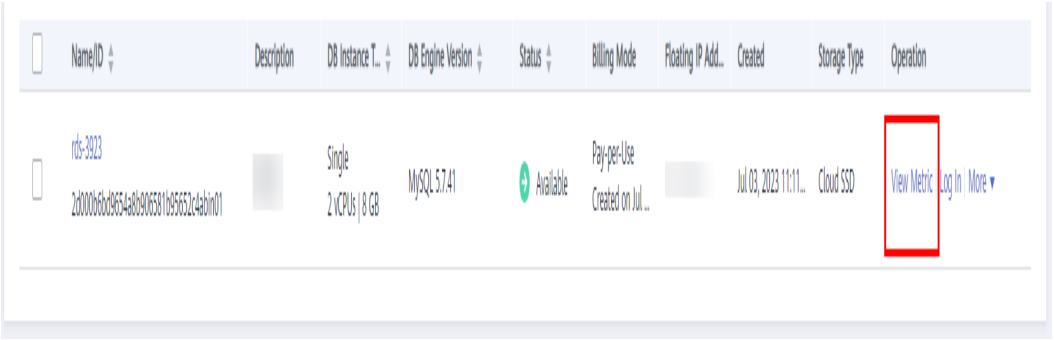
2. Configure long transaction alarms.

   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.

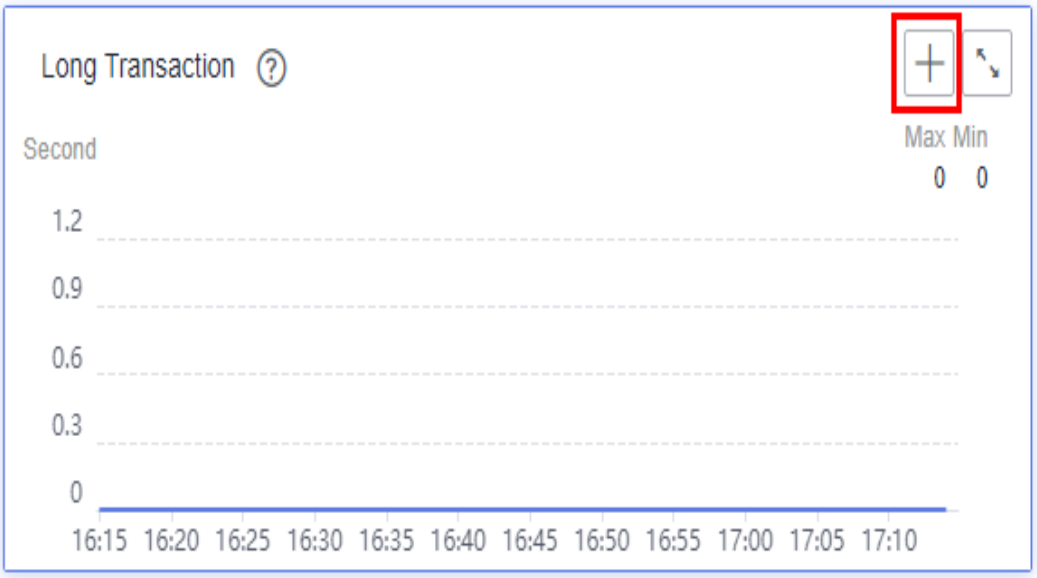
   c. View the **Long Transaction** metric.

Figure 9-2 Viewing metrics



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

Figure 9-3 Long Transaction



e. On the displayed page, set parameters as required. For details about the parameters, see **Creating an Alarm Rule**.

10 Configuring a Scheduled Event for an RDS for MySQL Instance

When you need to execute scheduled or periodic tasks in an RDS for MySQL instance, such as scheduled data synchronization, regular expired data deletion, and periodic data insertion, you can enable the event scheduler and configure a scheduled event in Data Admin Service (DAS) to automatically execute events defined in the instance based on the scheduled plan. This section describes how to use DAS to configure a scheduled event for an RDS for MySQL instance.

Constraints

- The event scheduler can be enabled only for DB instances running MySQL 5.6.43.2 or later, 5.7.25.2 or later, and 8.0.17.4 or later. If your DB engine version is beyond the above range, to use this function, **upgrade the minor version** first.
- The event scheduler cannot be enabled for read replicas.

Step 1: Enable the Event Scheduler

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 DB instance name.

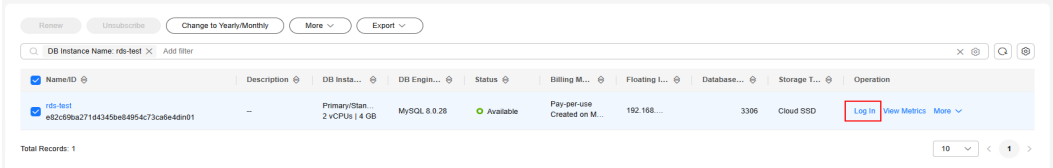
Step 5 On the **Overview** page, click **Enable** under **Event Scheduler**.

----End

Step 2: Configure a Scheduled Event

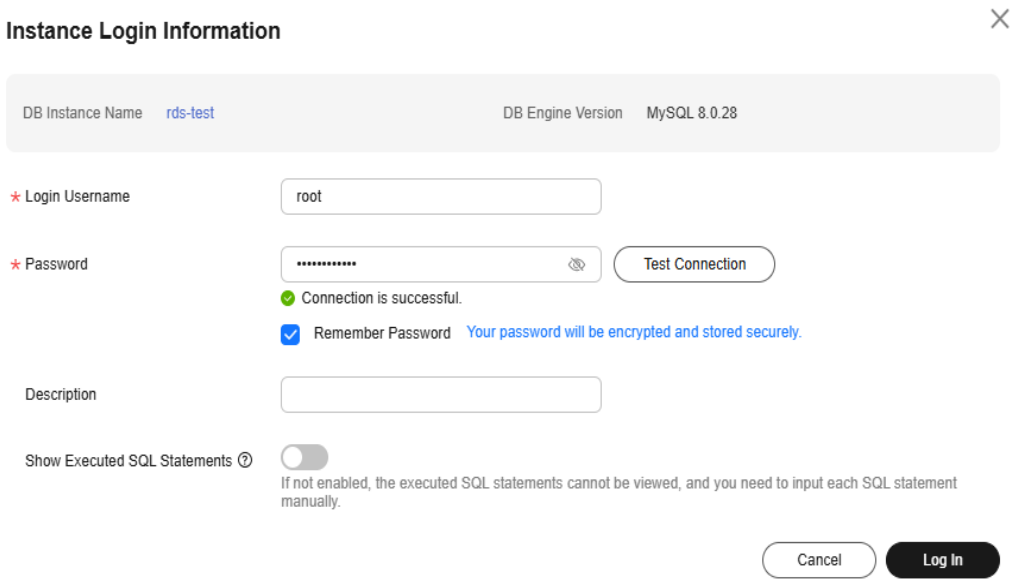
Step 1 On the **Instances** page, locate the DB instance and click **Log In** in the **Operation** column.

Figure 10-1 Logging in to an instance



Step 2 Enter **root** and its password, and click **Log In**.

Figure 10-2 Login page

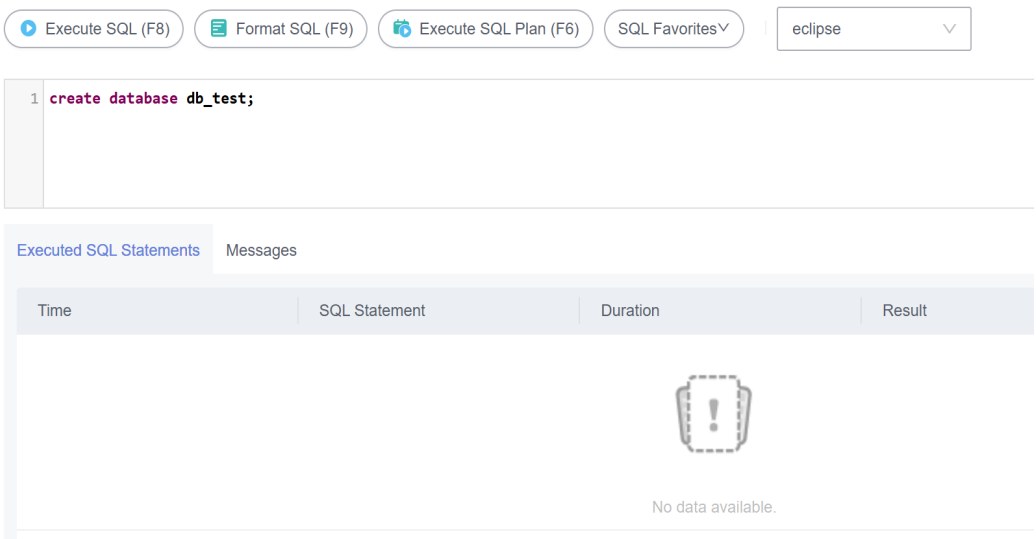


Step 3 Choose **SQL Operations** > **SQL Query**.

Step 4 In the SQL window, create a database named **db_test**.

```
create database db_test;
```

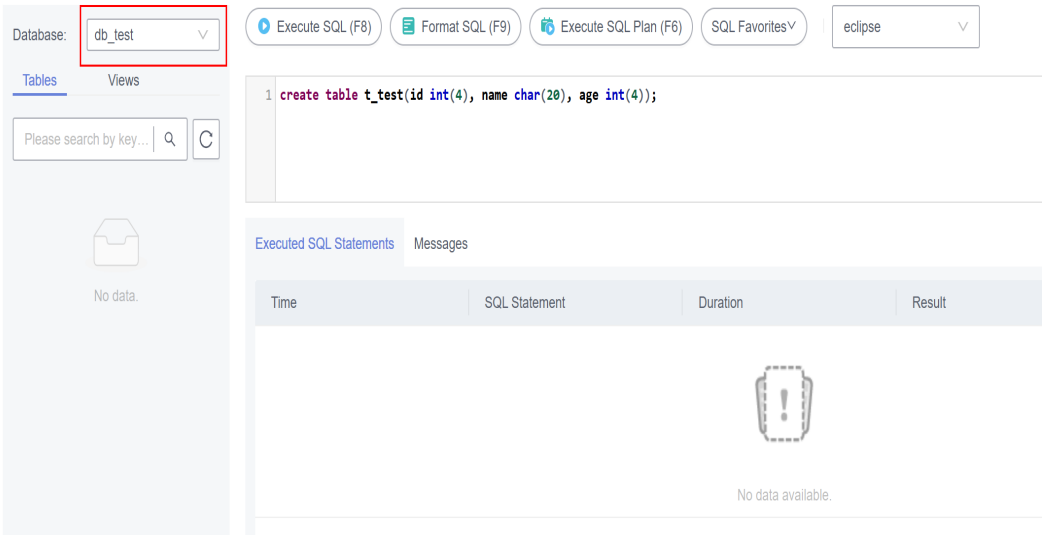
Figure 10-3 Creating a database



Step 5 Create a table named **t_test** in the **db_test** database.

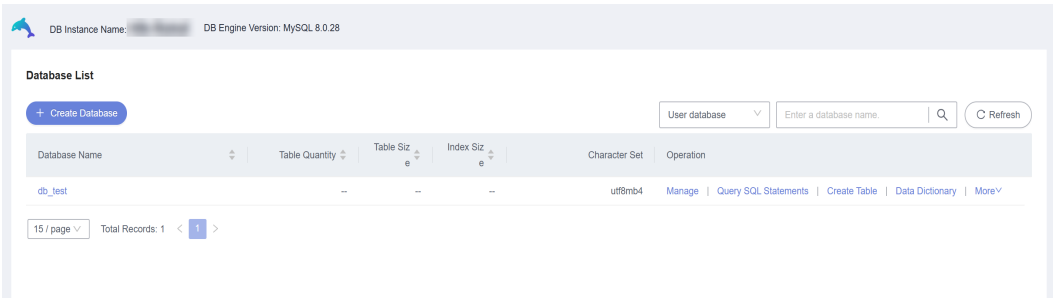
```
create table t_test(id int(4), name char(20), age int(4));
```

Figure 10-4 Creating a table



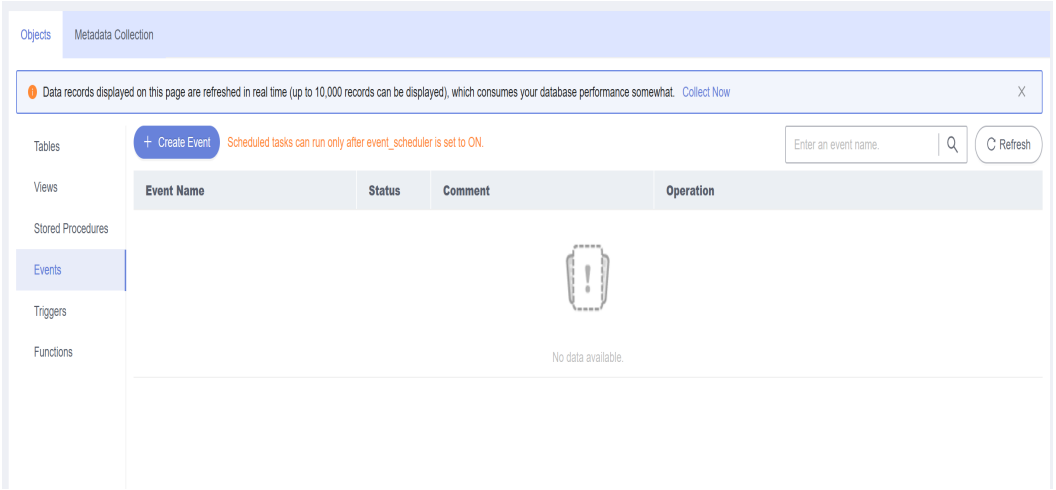
Step 6 On the homepage, click the database name.

Figure 10-5 Homepage



Step 7 On the displayed **Objects** page, choose **Events**. On the displayed page, click **Create Event**.

Figure 10-6 Objects



Step 8 Enter the event information and click **Create**.

Figure 10-7 Creating an event

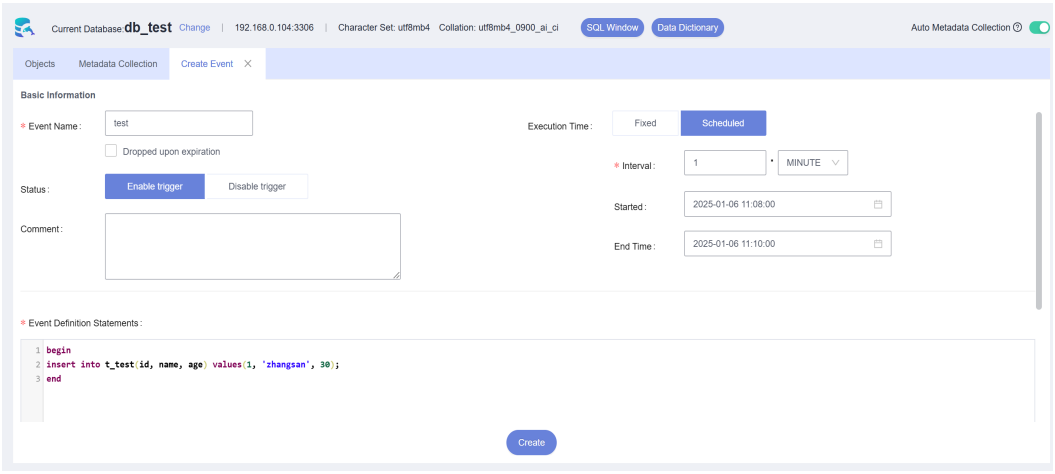


Table 10-1 Event description

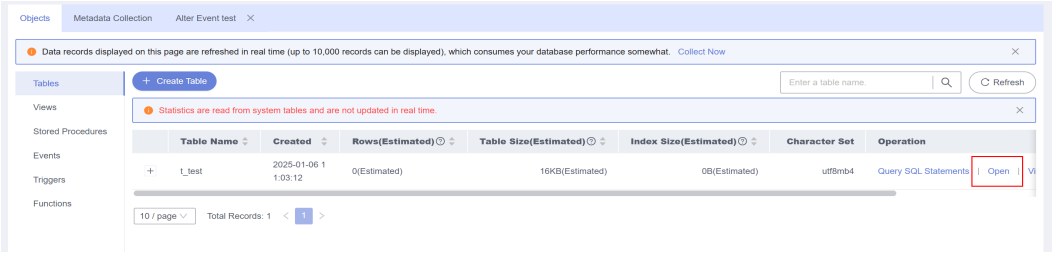
Parameter	Description
Event Name	Enter a custom event name.
Dropped upon expiration	• If this option is deselected, the event is always retained. • If you select this option, the event will be deleted upon expiration. – Events that are executed at a fixed time will be deleted once they are executed. – Events that are executed at a scheduled time will be deleted at the end time you specify.
Status	To execute an event, select **Enable trigger**.
Comment	Enter comments for the event.
Execution Time	• **Fixed** The event is executed only once at a fixed time. • **Scheduled** The event is executed at an interval you specify between the start time and end time. For example, an event is executed every minute between 09:50 and 10:00.

Event Definition Statements	Enter the statements to be executed when the event is triggered.
	For example, to insert a data record into the **t_test** table, enter the following statements:
	`begin`
	`insert into t_test(id, name, age) values(1, 'zhangsan', 30);`
	`end`

Step 9 In the displayed dialog box, click **Execute** to execute the event at the specified time.

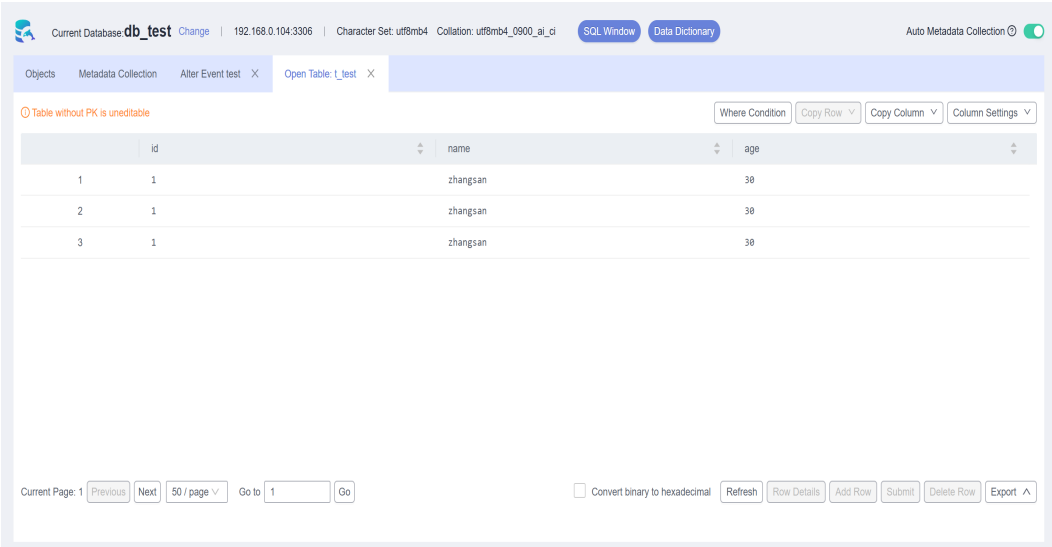
Step 10 In the object list, locate the table and click **Open**.

Figure 10-8 Opening a table



Step 11 Check the execution results of the event.

Figure 10-9 Checking execution results



----End

11 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)**.

12 MySQL Online DDL Tools

12.1 Introduction

In versions earlier than MySQL 5.6, DDL operations on the structure of a large table usually cause data manipulation language (DML) statements to be blocked and increase replication delay, so the database looks abnormal. This chapter introduces the DDL-based COPY and INPLACE algorithms of MySQL, the open-source tool gh-ost, and the INSTANT ADD COLUMN algorithm newly added in MySQL 8.0.

- The native COPY algorithm of MySQL adds a metadata write lock to the source table during data copy, causing DML statements to be blocked for a long time. This algorithm is no longer recommended.
- The INPLACE algorithm has great improvements over the COPY algorithm by making changes directly on the original table without generating temporary tables, so it occupies less space. In addition, the INPLACE operation holds metadata write locks for a short period of time, which does not cause long-term blocking of DML operations. However, modifying the structure of a large table still takes much time, and there will be a long replication delay when the standby instance replays the DDL statements.
- The open-source gh-ost splits a DDL operation into multiple small operations, reducing the time required for each operation to decrease replication delay. Reads and writes are briefly blocked only when the ghost table and original table are being renamed. gh-ost replays incremental data based on binlogs and maintains an extra heartbeat table to record the DDL execution process, supporting temporary suspension of the DDL process. gh-ost takes more time than the native DDL algorithm.
- The INSTANT ADD COLUMN algorithm proposed in MySQL 8.0 does not need to rebuild the entire table. It only records basic information about new columns in the metadata of the table. In this way, adding columns to a large table only takes several seconds. However, this algorithm applies only to a few DDL operations, such as adding columns, setting default values for columns, deleting default values from columns, and changing definitions of ENUM/SET columns.

Based on the characteristics of each algorithm and tool, you are advised to use the INSTANT algorithm to minimize the impact of DDL on your whole workload in

every possible case. In other cases, if your DB instance uses a primary/standby deployment or has read replicas and your workload is sensitive to replication delay, use gh-ost to perform DDL operations. If you need to quickly change a table structure and a short replication delay is acceptable, use INPLACE. The COPY algorithm, as it blocks DML operations for a long time, occupies a large amount of storage space, and takes a long time to execute, is not recommended when there is any other alternative.

Table 12-1 DDL tools

Item	MySQL COPY	MySQL INPLACE	gh-ost	INSTANT
Read operations during DDL execution	Allow	Allow	Allow	Allow
Write operations during DDL execution	Deny	Allow (deny for a short period of time)	Allow (deny for a short period of time)	Allow
Extra space occupied	Large	Small (slight increase if rebuild is required)	Large	Small
Execution duration	Very long	Long	Very long	Short
Replication delay	Long	Long	Short	Short

12.2 Native DDL Tools

COPY Algorithm

1. It creates a temporary table based on the original table definition.
2. It adds a write lock to the original table (DML is prohibited).
3. It executes DDL statements in the temporary table created in **1**.
4. It copies the data in the original table to the temporary table.
5. It releases the write lock of the original table.
6. It deletes the original table and renames the temporary table as the original table.

When the COPY algorithm is used, the table needs to be locked and DML write operations are forbidden. If **Lock** is set to **Shared**, read operations are allowed but write operations are not allowed. If **Lock** is set to **Exclusive**, both read and write operations are forbidden. This algorithm can be used in almost all DDL scenarios.

INPLACE Algorithm

INPLACE modifies the original table without generating a temporary table or copying data. There are two types:

- rebuild: The table needs to be rebuilt (with the clustered index reorganized), for example, optimizing a table, adding an index, adding or deleting a column, and modifying the NULL/NOT NULL attribute of a column.
- no-rebuild: The table does not need to be rebuilt. Only the metadata of the table needs to be modified, for example, deleting an index, changing a column name, changing the default value of a column, and changing the auto-increment value of a column.

If rebuild is used, the DML statements to be executed during DDL execution are cached. After the DDL operations are complete, the DML statements are applied to tables. Since metadata write locks will be degraded to metadata read locks during data copy, DML operations are almost not blocked during DDL execution.

Constraints on the INPLACE Algorithm

The INPLACE algorithm supports most DDL operations. But it has the following constraints, so in a few scenarios only the COPY algorithm can be used:

- It does not allow you to delete a primary key or add two primary keys at the same time.
- It does not allow you to change the data types of fields.
- It does not allow you to extend the length of the VARCHAR columns from less than 256 bits to more than 256 bits, because doing so will change the occupied space from 1 byte to 2 bytes. It does not allow you to reduce the length of the VARCHAR columns.
- It does not allow you to change the order of virtual columns or stored columns.
- It does not allow you to add foreign key constraints when **foreign_key_checks** is set to **1**.
- It does not allow you to partition tables, or optimize or delete partitions.

12.3 gh-ost

Context

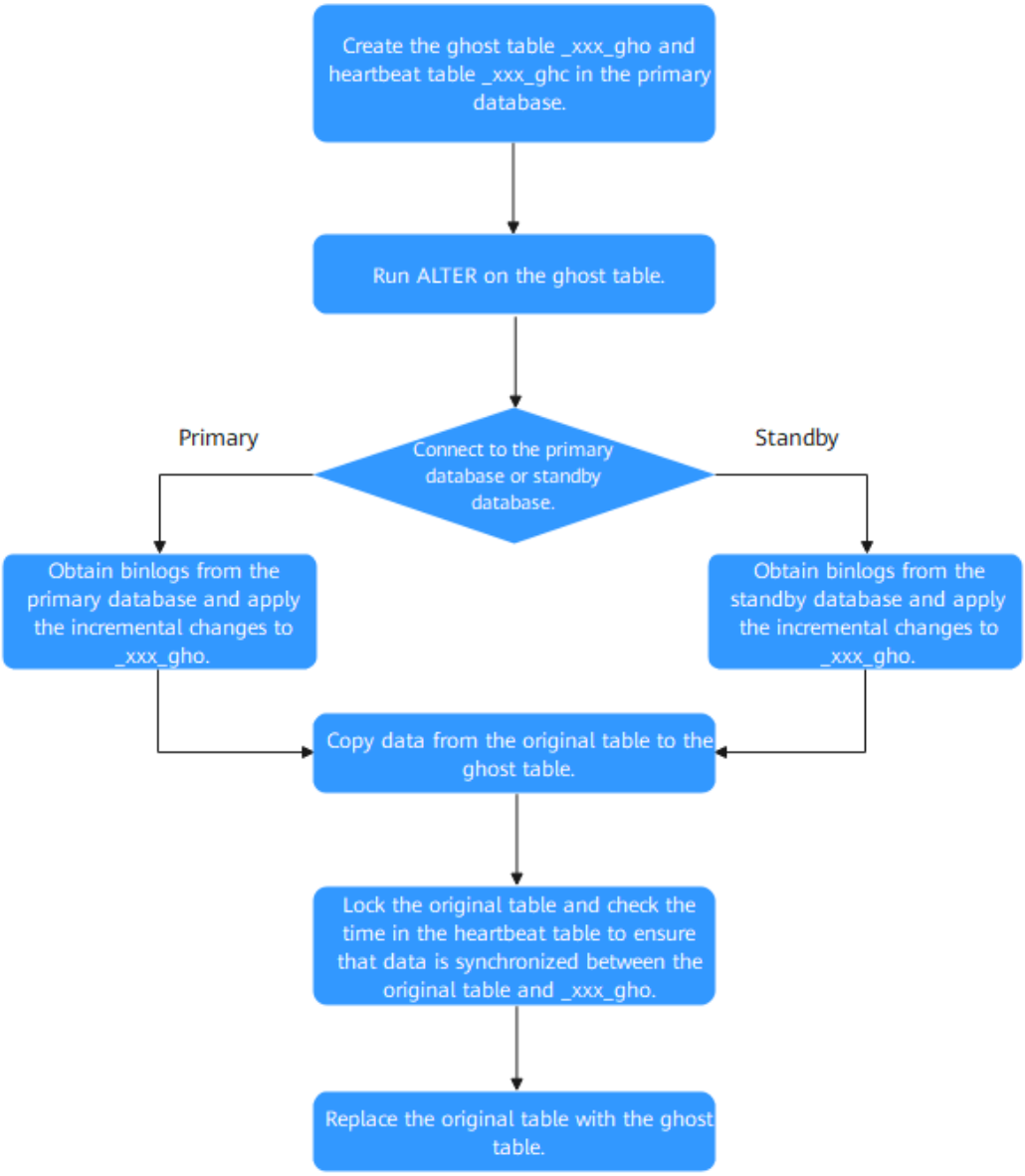
Percona offers an open-source DDL tool, pt-osc. It executes operations by using triggers to copy rows from the original table to the new table. Using triggers can speed up synchronization but causes a large overhead, affecting the performance of the primary database. In addition, data copies and data changes may be processed concurrently. If a table is frequently updated during migration, a large number of lock contention problems may occur.

gh-ost is an open-source online DDL tool provided by GitHub. Unlike pt-osc, gh-ost does not depend on triggers. Instead, it simulates the standby database to obtain incremental changes from binlogs in the row format and asynchronously applies the changes to the ghost table. It decouples the migration's write load from the workload of the primary server, avoiding the impact on the performance

of the primary database. Asynchronously applying incremental changes also avoids lock contention caused by triggers. In addition, gh-ost maintains a heartbeat table to record each phase in the DDL process. When an exception occurs, data can be restored to the specified position based on the heartbeat log. This solves the problem that pt-osc needs to start from the beginning when an exception occurs.

Process

Figure 12-1 Flowchart



Three gh-ost Modes

- (Default mode) gh-ost connects to the standby database and performs a cutover in the primary database.
  - In the primary database, gh-ost creates the **_xxx_gho** table with the same structure as the original table and the **_xxx_ghc** table that records

the change status. The **_xxx_ghc** table is used to write the progress and time of online DDL operations.
  - The structure of the **_xxx_gho** table is modified.
  - The existing data of the original table is copied to **_xxx_gho** in the primary database.
  - The incremental binlogs are obtained from the standby database and the incremental changes are applied to **_xxx_gho**.
  - The original table is locked and the time in the **_xxx_ghc** table is checked to ensure that data is synchronized between the original table and **_xxx_gho**.
  - The original table is replaced with **_xxx_gho**.
- gh-ost connects to the primary database and performs a cutover in the primary database.
  - The **_xxx_gho** and **_xxx_ghc** tables are created in the primary database.
  - The structure of the **_xxx_gho** table is modified.
  - The existing data of the original table is copied to **_xxx_gho** in the primary database.
  - The incremental binlogs are obtained from the primary database and the incremental changes are applied to **_xxx_gho**.
  - The original table is locked and the time in the **_xxx_ghc** table is checked to ensure that data is synchronized between the original table and **_xxx_gho**.
  - The original table is replaced with **_xxx_gho**.
- gh-ost performs a test and cutover on the standby database.

  In this mode, gh-ost connects to the primary database. However, all operations are performed on the standby database and no modification is made to the primary database.

  -migrate-on-replica means that gh-ost directly migrates the table on the standby database. It performs the cutover when replication of the standby database is running.

  -test-on-replica indicates that the migration is only for test purposes. Replication is stopped before a cutover is performed. The original table and temporary table are swapped and then swapped back. The original table returns to its original place. Both of the tables are left with replication stopped. You may examine the two and compare data.

Common Parameters

For details about gh-ost parameters, see the **official documentation**.

Constraints

- Row-based binlogs must be used, and the value of **binlog_row_image** must be **FULL**.
- The required user permissions include SUPER, REPLICATION CLIENT, and REPLICATION SLAVE.
  - If the binlogs are in the row format, you can add **-assume-rbr**. In this case, the SUPER permission is not required.

- Tables with foreign key constraints are not supported.
- Tables with triggers are not supported.
- The tables before and after DDL execution must have the same primary key or non-null unique indexes.
- If the primary key or non-null unique index of a table to be migrated contains enumeration types, the migration efficiency will be greatly reduced.

Example

```
gh-ost -max-load=Threads_running=20 \
        -critical-load=Threads_running=100 \
        -chunk-size=2000 -user="temp"
        -password="test" -host=**.*.*.* \
        -allow-on-master -database="sbtest" -table="sbtest1" \
        -alter="engine=innodb" -cut-over=default \
        -exact-rowcount -concurrent-rowcount -default-retries=120 \
        -timestamp-old-table -assume-rbr -panic-flag-file=/tmp/ghost.panic.flag \
        -execute
```

12.4 INSTANT ADD COLUMN

Context

Generally, DDL operations on large tables have great impact on workloads. They need to be performed during off-peak hours. MySQL 5.7 supports the native DDL tool COPY and INPLACE algorithms and the open-source DDL tool gh-ost, reducing blocked DML operations during DDL execution. But it still takes a long time to perform DDL operations on large tables.

INSTANT ADD COLUMN eliminates the need to rebuild the entire table when adding columns. It only needs to record the basic information about the new columns in the table metadata. However, only a limited number of DDL operations are supported.

Syntax

If **ALGORITHM=INSTANT** is added to the end of the ALTER statement, the INSTANT algorithm is used. Here is an example:

```
ALTER TABLE *tbl_name* ADD COLUMN *column_name* *column_definition*,
ALGORITHM=INSTANT;
```

Constraints

This algorithm can only be used when you:

- Add, delete, or rename columns (for versions later than MySQL 8.0.28) in certain scenarios.
- Set or delete the default value of a column.
- Modify the definition of the ENUM or SET column.
- Change the index type (B-Tree | hash).
- Add or delete a virtual column.

- Rename a table.

Constraints on adding or deleting columns:

- An ALTER TABLE statement cannot combine the addition of a column with other actions that do not support the INSTANT algorithm.
- New columns are placed at the end and the column sequence cannot be changed. (In versions later than MySQL 8.0.29, columns can be added to any position.)
- Columns cannot be quickly added to or deleted from a table whose row format is COMPRESSED.
- Columns cannot be quickly added to or deleted from a table that has a full-text index.
- Columns cannot be quickly added to or deleted from a temporary table.

Constraints on renaming columns:

- Columns referenced by other tables cannot be renamed.
- The operation of renaming a column and the operation of generating or deleting a virtual column cannot be in the same statement.

Constraints on modifying the ENUM or SET column.

- The storage space occupied by the ENUM or SET column data type cannot be changed.

Constraints on adding or deleting virtual columns:

- Virtual columns cannot be added to or deleted from partitioned tables.

New Data Dictionary Information

When INSTANT ADD COLUMN is executed, MySQL saves the number of fields before INSTANT ADD COLUMN is executed for the first time and the default value of the column added each time to the **se_private_data** field in the **tables** system table.

- **dd::Table::se_private_data::instant_col**: indicates the number of columns in the table before INSTANT ADD COLUMN is executed for the first time.
- **dd::Column::se_private_data::default_null**: indicates whether the default value of the instant column is **NULL**.
- **dd::Column::se_private_data::default**: indicates the default value stored when the default value of the instant column is not **NULL**.

Importing Data Dictionary

When MySQL reads table definitions from system tables, it loads instant column information to the InnoDB table object **dict_table_t** and index object **dict_index_t**.

- **dict_table_t::n_instant_cols**: indicates the number of non-virtual fields (including system columns) before INSTANT ADD COLUMN is executed for the first time.
- **dict_index_t::instant_cols**: indicates whether there is an instant column.

- **dict_index_t::n_instant_nullable**: indicates the number of fields that can be null before INSTANT ADD COLUMN is executed for the first time.
- **dict_col_t::instant_default**: indicates the default value and length of the instant column.

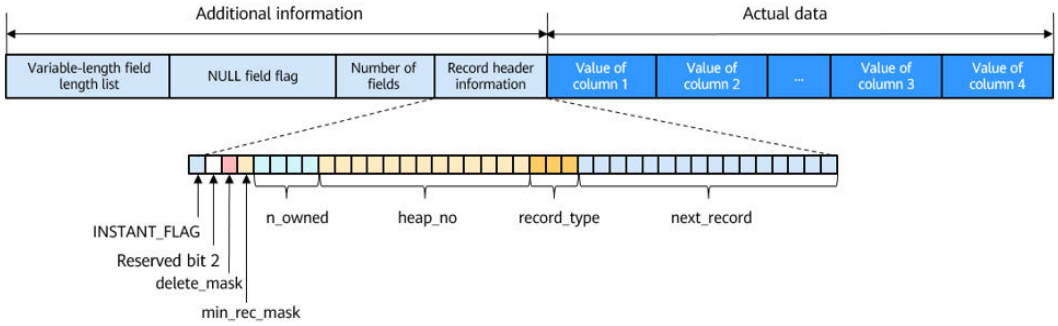
Record Format

To support INSTANT ADD COLUMN, a new record format is introduced for the COMPACT and DYNAMIC types to record the number of fields.

- If INSTANT ADD COLUMN has not been performed, the row record format of the table remains unchanged.
- If INSTANT ADD COLUMN has been performed, a special flag is set for each new record, and the number of fields is stored in the records.

INSTANT_FLAG uses a bit in info bits. If a record is inserted after the first execution of INSTANT ADD COLUMN, the flag is set to **1**.

Figure 12-2 Record format



Query

The query process remains unchanged. For instant columns that are not stored in records, use the default value.

Insertion

After INSTANT ADD COLUMN is executed, the format of the old data does not change and the newly inserted data is stored in the new format. If a bit in the info bits of a new record is set to **REC_INFO_INSTANT_FLAG**, the record is created after INSTANT ADD COLUMN is executed.

12.5 DDL Tool Test Comparison

Test Procedure

1. Create four tables. The table structures are as follows:

```
CREATE TABLE if not exists users
  (
   `rid` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
    `nid` bigint(20) DEFAULT NULL,
    `level` int(11) DEFAULT NULL,
```



```
  `vip` int(11) DEFAULT NULL,
  `vip_exp` int(11) DEFAULT NULL,
  `reg_channel` int(11) DEFAULT NULL,
  `guild_id` bigint(20) unsigned DEFAULT '0',
  `guild_open` tinyint(1) DEFAULT '0',
  `forbid_login_time` bigint(20) DEFAULT NULL,
  `forbid_talk_time` bigint(20) DEFAULT NULL,
  `ctime` bigint(20) DEFAULT NULL,
  `mtime` datetime(3) DEFAULT NULL,
  `last_offline_time` bigint(20) DEFAULT NULL,
  `friend_open` tinyint(1) DEFAULT '0',
  `user_data_str` mediumblob,
  `name` varchar(64) DEFAULT NULL,
  `db_fix_version` int(10) DEFAULT '0',
  PRIMARY KEY (`rid`),
  KEY `idx_users_99_nid` (`nid`),
  KEY `idx_users_99_level` (`level`),
  KEY `idx_users_99_ctime` (`ctime`),
  KEY `idx_users_99_mtime` (`mtime`),
  KEY `idx_users_99_last_offline_time` (`last_offline_time`),
  KEY `idx_users_99_name` (`name`)
) ENGINE=InnoDB AUTO_INCREMENT=4393751571200 DEFAULT CHARSET=utf8mb4;
```

2. Insert 30 million rows of data into each table.
3. Use the MySQL native COPY algorithm to add a column to table 1. During the execution, create a new session and perform the SELECT, UPDATE, and INSERT operations on 100,000 data records.
4. Use the MySQL native INPLACE algorithm to add a column to table 2. During the execution, create a new session and perform the SELECT, UPDATE, and INSERT operations on 100,000 data records.
5. Use gh-ost to add a column to table 3. During the execution, create a new session and perform the SELECT, UPDATE, and INSERT operations on 100,000 data records.
6. Record the execution durations of DDL and DML statements.

Table 12-2 Test data (unit: s)

Operation	MySQL COPY	MySQL INPLACE	gh-ost
Adding a column	1294.29	755.52	1876.79
SELECT	1.35	1.29	1.29
UPDATE	1266.78	0.19	0.11
INSERT	1296.19	7.47	4.49

Test Results

1. MySQL COPY: The UPDATE and INSERT statements are blocked, but the SELECT statement is executed properly.
2. MySQL INPLACE: DML statements are not blocked for a long time, and adding a column to a large table takes the shortest time.
3. gh-ost: It almost does not block DML statements. It takes a longer time to add a column than the two native MySQL algorithms.

Suggestions

INPLACE blocks DML operations only for a short time while performing DDL operations. If you have no strict requirements on the primary/standby replication delay, you are advised to use this algorithm to quickly change the table structure. If your workload is sensitive to primary/standby replication delay, gh-ost is recommended. If you use MySQL 8.0.12 or later and the INSTANT algorithm conditions are met, you can use INSTANT to minimize the impact on workloads.

