ALIBABA CLOUD

Alibaba Cloud

Apsara Stack Enterprise

ApsaraDB for RDS

ApsaraDB RDS for PostgreSQL User Guide

Product Version: v3.16.2

Document Version: 20220913

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	Notice: If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.What is ApsaraDB RDS?

ApsaraDB RDS is a stable, reliable, and scalable online database service. Based on the high-performance storage, ApsaraDB RDS provides a set of solutions for disaster recovery, backup, restoration, monitoring, and migration.

ApsaraDB RDS supports four database engines: MySQL, SQL Server, PolarDB, and PostgreSQL. You can create database instances based on these database engines to meet your business requirements. This topic describes the PostgreSQL engine.

ApsaraDB RDS for PostgreSQL

ApsaraDB RDS for PostgreSQL is developed based on the advanced open source database. It is fully compatible with SQL and supports a diverse range of data such as JSON, IP, and geometric data. In addition to features such as transactions, subqueries, multi-version concurrency control (MVCC), and data integrity check, ApsaraDB RDS for PostgreSQL integrates a series of features including high availability, backup, and restoration to ease O&M loads.

2.Limits on ApsaraDB RDS for PostgreSQL

Before you use ApsaraDB RDS for PostgreSQL, you must understand its limits and take the necessary precautions.

The following table describes the limits on ApsaraDB RDS for PostgreSQL.

Operation	Limit
Root permissions of databases	Superuser permissions are not provided.
Database replication	ApsaraDB RDS for PostgreSQL provides a primary/secondary replication architecture except in the Basic Edition. The secondary instances in the architecture are hidden and cannot be accessed by your applications.
Instance restart	Instances must be restarted by using the ApsaraDB RDS console or API operations.

3.Log on to the ApsaraDB RDS console

This topic describes how to log on to the ApsaraDB RDS console.

Prerequisites

- Before you log on to the Apsara Uni-manager Management Console, the endpoint of the console is obtained from the deployment staff.
- We recommend that you use Google Chrome.

Procedure

1. Enter your username and password.

   Obtain the username and password that you use to log on to the Apsara Uni-manager Management Console from the operations administrator.

   > **Note** The first time that you log on to the Apsara Uni-manager Management Console, you must change the password of your account. For security purposes, your password must meet the minimum complexity requirements. The password must be 10 to 32 characters in length and must contain at least two of the following character types:
   >
   > - Uppercase or lowercase letters
   > - Digits
   > - Special characters, including exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%)

2. Click **Log On**.
3. If your account has multi-factor authentication (MFA) enabled, perform corresponding operations in the following scenarios:
   - You log on to the Apsara Uni-manager Management Console for the first time after MFA is enabled by the administrator:
     a. On the Bind Virtual MFA Device page, bind an MFA device.
     b. Enter the username and password again as in Step 2 and click **Log On**.
     c. Enter a six-digit MFA verification code and click **Authenticate**.
   - You have enabled MFA and bound an MFA device:

     Enter a six-digit MFA verification code and click **Authenticate**.

     > **Note** For more information, see the *Bind a virtual MFA device to enable MFA* topic in *Apsara Uni-manager Management Console User Guide*.

4. In the top navigation bar, choose **Products > Database Services > ApsaraDB RDS**.

4.Quick Start

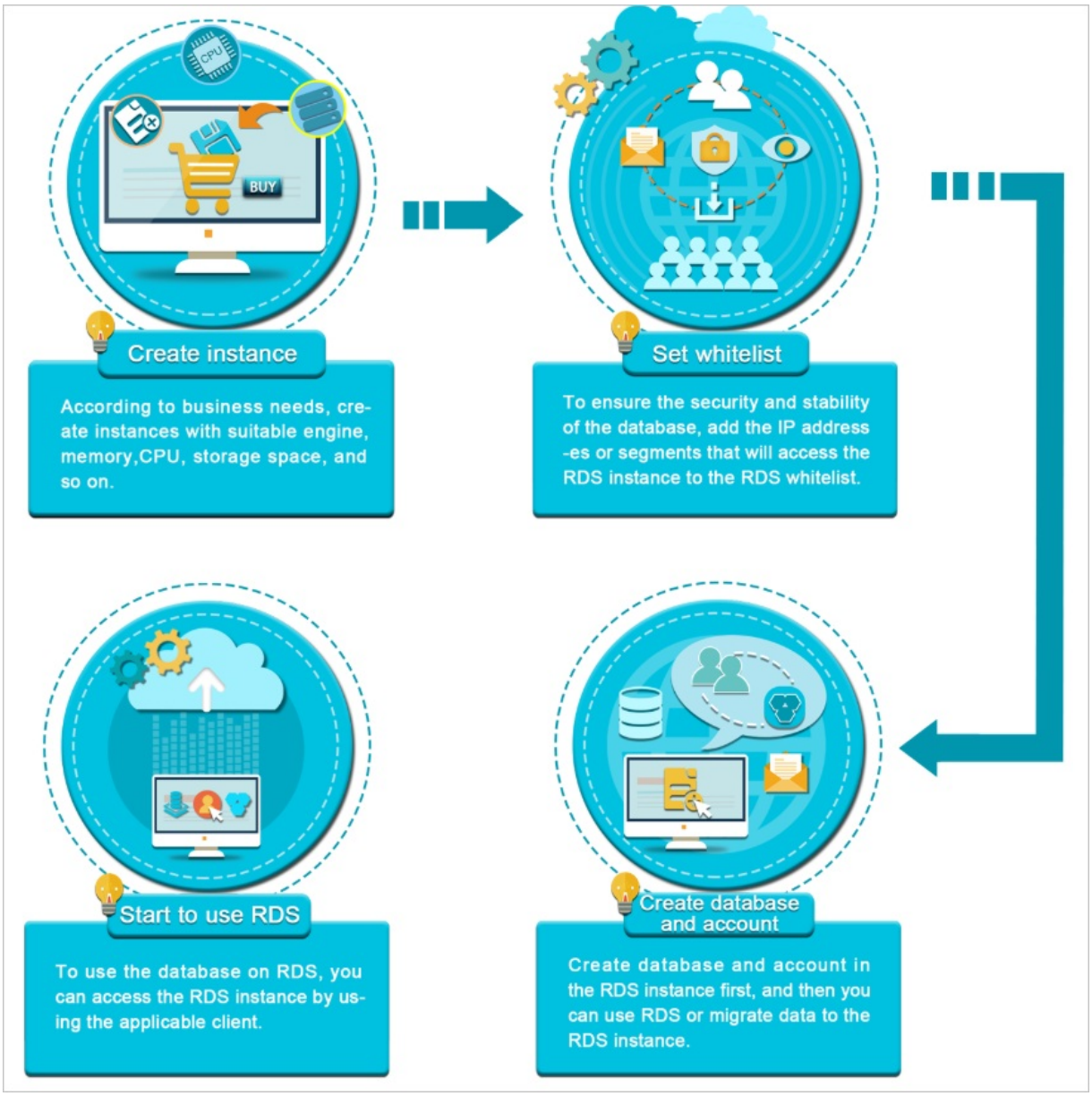
4.1. Procedure

ApsaraDB RDS quick start covers the following topics: creating an ApsaraDB RDS instance, configuring a whitelist, creating a database, creating an account, and connecting to the instance.

Flowchart for an ApsaraDB RDS instance

If you are using ApsaraDB RDS for the first time, you can start with Limits.

The following figure shows the operations that you must perform before you use an ApsaraDB RDS instance.



4.2. Create an instance

This topic describes how to create an ApsaraDB RDS for PostgreSQL instance in the ApsaraDB RDS console.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, click **Create Instance** in the upper-right corner.
3. Configure the parameters described in the following table.

Section	Parameter	Description
Basic Configurations	Organization	The organization to which the instance belongs.
	Resource Set	The resource set to which the instance belongs.
Area	Region	The region in which you want to create the instance. Services in different regions cannot communicate over an internal network. After the instance is created, the region cannot be changed.
	Primary Node Zone	The zone in which the primary instance is deployed.
	Deployment Method	Specifies whether to deploy the primary and secondary instances in separate zones. ApsaraDB RDS supports **Multi-zone Deployment** and **Single-zone Deployment**. If you select **Multi-zone Deployment**, you must configure **Secondary Node Zone**.
	Secondary Node Zone	The zone in which the secondary instance is deployed. This parameter is available only when you set the **Deployment Method** parameter to **Multi-zone Deployment**. **Note** If you select the same zone for both the primary and secondary instances, the deployment is equivalent to single-zone deployment.
	Database Engine	The database engine of the instance. Select **PostgreSQL**.
	Engine Version	The version of the database engine. Valid values: ○ 10.0 ○ 11.0 ○ 12.0 ○ 13.0
	Edition	The edition of the instance. Select **High-availability Edition**.

Section	Parameter	Description
Specifications	Storage Type	The storage type of the instance. Local SSDs and standard SSDs are supported. The storage types that are supported vary based on the engine version: ○ PostgreSQL 10.0: Local SSDs and standard SSDs are supported. ○ PostgreSQL 11.0: Standard SSDs are supported. ○ PostgreSQL 12.0: Local SSDs are supported. ○ PostgreSQL 13.0: Standard SSDs are supported.
	Encrypted	Specifies whether to encrypt standard SSDs. This parameter is available only when you set the **Storage Type** parameter to **Standard SSD**. If you select Encrypted, you must specify the **Encryption Key** parameter. If you do not have a key, you must first create one in the Key Management Service (KMS) console. For more information, see Create a CMK in *Key Management Service User Guide*.
	Encryption Key	The key that is used to encrypt standard SSDs. This parameter is available only when you select **Encrypted**.
	Instance Specifications	The specifications of the instance. The maximum number of connections and the maximum IOPS vary based on the memory size. The actual specifications are displayed in the console. For more information, see Instance types in *ApsaraDB RDS Product Introduction*.
	Storage Capacity	The storage capacity that is provided to store data files, system files, binlog files, and transaction files in the instance. Valid values: 20 to 6000. Unit: GB. The value must be in 1 GB increments.
Network	Connection Type	The connection type of the instance. Valid values: ○ **Internet**: Instances of this connection type can be connected over the Internet. ○ **Internal Network**: Instances of this connection type can be connected over an internal network. **Note** After the instance is created, the value of this parameter cannot be changed. Proceed with caution.

Section	Parameter	Description
	Network Type	The network type of the instance. Valid values: ○ **Classic Network**: Cloud services in the classic network are not isolated. Unauthorized access to a cloud service is blocked only by the security group or whitelist policy of the service. ○ **VPC**: A virtual private cloud (VPC) helps you build an isolated network environment on Apsara Stack. You can customize route tables, CIDR blocks, and gateways in a VPC. We recommend that you select VPC for improved security. **Note** ■ If you configure multi-zone deployment, you must create vSwitches for the zones of primary and secondary instances in the specified VPC. ■ If you select VPC, you must specify a VPC and a vSwitch.
	IP Whitelist	The IP address or CIDR block that is allowed to connect to the instance.
Instance Settings	**Quantity**	The number of instances that you want to create. Default value: 1.
	Instance Name	The name of the instance. ○ The name must be 2 to 64 characters in length. ○ The name must start with a letter. ○ The name can contain letters, digits, and the following special characters: _ - : ○ The name cannot start with http:// or https://.

4. Click **Submit**.

4.3. Configure an IP address whitelist

This topic describes how to configure a whitelist for an ApsaraDB RDS instance. Only entities that are listed in a whitelist can access your ApsaraDB RDS instance.

Context

Whitelists make your ApsaraDB RDS instance more secure and do not interrupt the operations of your ApsaraDB RDS instance when you configure whitelists. We recommend that you perform maintenance on your whitelists on a regular basis.

To configure a whitelist, perform the following operations:

- Configure a whitelist: Add IP addresses to allow them to connect to the ApsaraDB RDS instance.

> **Note** The IP address whitelist labeled default contains only the default IP address 0.0.0.0/0, which allows all entities to access your ApsaraDB RDS instance.

- Configure an ECS security group: Add an ECS security group for the ApsaraDB RDS instance to allow ECS instances in the group to connect to the ApsaraDB RDS instance.

Procedure

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Data Security**.
5. On the **Whitelist Settings** tab, click **Edit** corresponding to the **default** whitelist.

   > **Note** You can also click **Create Whitelist** to create a whitelist.

6. In the **Edit Whitelist** dialog box, enter the IP addresses or CIDR blocks used to access the instance and click **OK**. The following section describes the rules:
   - If you enter the CIDR block 10.10.10.0/24 in the IP Addresses field, all IP addresses in the 10.10.10.X format can access your ApsaraDB RDS instance.
   - If you enter more than one IP address or CIDR block, you must separate them with commas (,). Do not add spaces before or after the commas. Example: 192.168.0.1,172.16.213.9.
   - If you click **Add Internal IP Addresses of ECS Instances**, the IP addresses of all ECS instances created within your Alibaba Cloud account are displayed. You can select the required IP addresses to add them to the IP address whitelist.

4.4. Create a database and an account

Before you start to use ApsaraDB RDS, you must create databases and accounts on an ApsaraDB RDS instance. This topic describes how to create a database and an account on an ApsaraDB RDS for PostgreSQL instance.

Account types

ApsaraDB RDS for PostgreSQL instances support two types of accounts: privileged accounts and standard accounts. The following table describes these account types.

Account type	Description

Account type	Description
Privileged account	• You can create and manage privileged accounts only by using the ApsaraDB RDS console or API operations. • If your ApsaraDB RDS instance uses local SSDs, you can create only a single privileged account. If your ApsaraDB RDS instance uses standard or enhanced SSDs, you can create more than one privileged account. A privileged account allows you to manage all the standard accounts and databases that are created on your ApsaraDB RDS instance. • A privileged account has more permissions that allow you to manage your ApsaraDB RDS instance at more fine-grained levels. For example, you can grant the query permissions on different tables to different users. • A privileged account has the permissions to disconnect accounts that are created on your ApsaraDB RDS instance.
Standard account	• You can create and manage standard accounts by using the ApsaraDB RDS console, API operations, or SQL statements. • You can create more than one standard account on your ApsaraDB RDS instance. • You must grant the permissions on specific databases to a standard account. • A standard account does not have the permissions to create, manage, or disconnect other accounts on your ApsaraDB RDS instance.

Precautions

- If your ApsaraDB RDS instance uses local SSDs, you can create one privileged account in the ApsaraDB RDS console. After the privileged account is created, it cannot be deleted. You can also create and manage more than one standard account by using SQL statements.
- If your ApsaraDB RDS instance uses standard or enhanced SSDs, you can create more than one privileged account and standard account in the ApsaraDB RDS console. You can also create and manage more than one standard account by using SQL statements.
- To migrate data from an on-premises database to your ApsaraDB RDS instance, you must create a database and an account on the ApsaraDB RDS instance. Make sure that the created database has the same properties as the on-premises database. Also make sure that the created account has the same permissions on the created database as the account that is authorized to manage the on-premises database.
- Follow the least privilege principle to create accounts and grant them read-only permissions or read and write permissions on databases. If necessary, you can create more than one account and grant them only the permissions on specific databases. If an account does not need to write data to a database, grant only the read-only permissions on that database to the account.
- For security purposes, we recommend that you specify strong passwords for the accounts on your ApsaraDB RDS instance and change the passwords on a regular basis.

Create a privileged account on an instance that uses local SSDs

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.

4. In the left-side navigation pane, click **Accounts**.
5. On the Accounts page, click **Create Privileged Account** and configure the following parameters.

Parameter	Description
Database Account	○ The name of the account must be 2 to 16 characters in length. ○ The name can contain lowercase letters, digits, and underscores (_). ○ The name must start with a letter and end with a letter or digit.
Password	○ The password of the account must be 8 to 32 characters in length. ○ The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. ○ Special characters include ! @#$%^&*()_+-=
Re-enter Password	Enter the password of the account again.

6. Click **Create**.

Create a privileged or standard account on an instance that uses standard or enhanced SSDs

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Accounts**.
5. On the Accounts page, click **Create Account** and configure the following parameters.

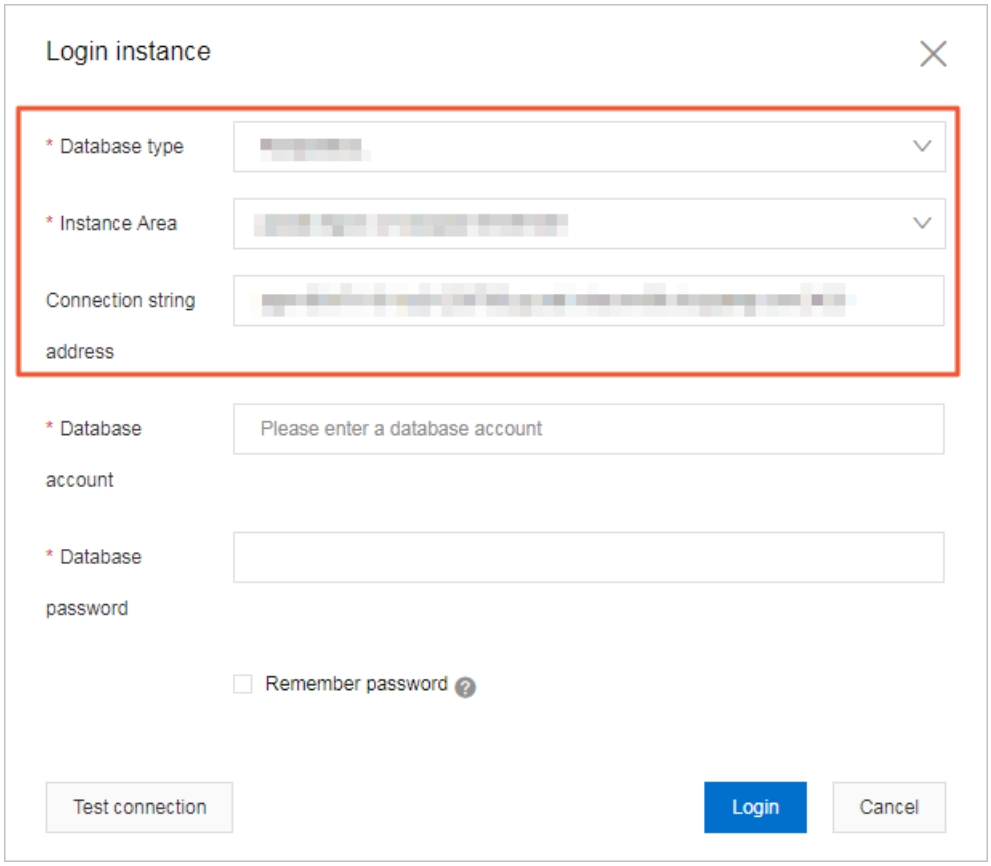
Parameter	Description
Database Account	○ The name of the account must be 2 to 16 characters in length. ○ The name can contain lowercase letters, digits, and underscores (_). ○ The name must start with a letter and end with a letter or digit.
Account Type	Select **Privileged Account** or **Standard Account**.
Password	○ The password of the account must be 8 to 32 characters in length. ○ The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. ○ Special characters include ! @#$%^&*()_+-=
Re-enter Password	Enter the password of the account again.

Parameter	Description
Description	This parameter is optional. You can enter relevant description to make the instance identifiable. The description can be up to 256 characters in length.

6. Click **Create**.

Create a database and a standard account

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. Click **Log On to DB** in the upper-right corner of the page.
5. In the **Login instance** dialog box of the **DMS** console, check values of **Database type**, **Instance Area**, and **Connection string address**. If the information is correct, enter **Database account** and **Database password**, as shown in the following figure.



Parameter	Description
Database type	The engine of the database. By default, the engine of the database to be connected is displayed.
Instance Area	The region where the instance is deployed. By default, the region of the current instance is displayed.

Parameter	Description
Connection string address	The endpoint of the instance. By default, the endpoint of the current instance is displayed.
Database account	The account of the database to be connected.
Database password	The password of the account used to connect to the database.

6. Click **Login**. If you want the browser to remember the password, select **Remember password** before you click **Login**.

   > **Note** If a message prompts you that the connection fails, the problem may be caused by an improperly configured whitelist. Reconfigure the whitelist in the console. For more information, see Configure an IP address whitelist.

7. The **SQLConsole** page appears after you log on to the instance. Execute a statement in the following format to create a database:

```
CREATE DATABASE name
 [ [ WITH ] [ OWNER [=] user_name ]
        [ TEMPLATE [=] template ]
        [ ENCODING [=] encoding ]
        [ LC_COLLATE [=] lc_collate ]
        [ LC_CTYPE [=] lc_ctype ]
        [ TABLESPACE [=] tablespace_name ]
        [ CONNECTION LIMIT [=] connlimit ] ]
```

   For example, if you want to create a database named test, execute the following statement:

```
create database test;
```

8. Click **execute**.
9. In the SQL window, execute a statement in the following format to create a standard account:

```
CREATE USER name [ [ WITH ] option [ ... ] ]
where option can be:
    SUPERUSER | NOSUPERUSER
  | CREATEDB | NOCREATEDB
  | CREATEROLE | NOCREATEROLE
  | CREATEUSER | NOCREATEUSER
  | INHERIT | NOINHERIT
  | LOGIN | NOLOGIN
  | REPLICATION | NOREPLICATION
  | CONNECTION LIMIT connlimit
  | [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
  | VALID UNTIL 'timestamp'
  | IN ROLE role_name [, ...]
  | IN GROUP role_name [, ...]
  | ROLE role_name [, ...]
  | ADMIN role_name [, ...]
  | USER role_name [, ...]
  | SYSID uid
```

For example, if you want to create a user account named test2 whose password is 123456, execute the following statement:

```
create user test2 password '123456';
```

10. Click **execute**.

4.5. Connect to an ApsaraDB RDS for PostgreSQL instance

This topic describes how to use Data Management (DMS) or the pgAdmin 4 client to connect to an ApsaraDB RDS instance.

Context

You can log on to DMS from the ApsaraDB RDS console and then connect to an ApsaraDB RDS instance.

DMS is a data management service that integrates data, schema, and server management, access security, BI charts, data trends, data tracking, and performance optimization. DMS can be used to manage relational and non-relational databases, such as MySQL, SQL Server, PostgreSQL, MongoDB, and Redis. It can also be used to manage Linux servers.

You can also use a client to connect to an ApsaraDB RDS instance. ApsaraDB RDS for PostgreSQL is fully compatible with PostgreSQL. You can connect to an ApsaraDB RDS for PostgreSQL instance in a similar manner as you would connect to an open source PostgreSQL instance. In this topic, the pgAdmin 4 client is used to connect to an ApsaraDB RDS instance.

Use DMS to connect to an ApsaraDB RDS instance

For more information about how to use DMS to connect to an ApsaraDB RDS instance, see Log on to an ApsaraDB for RDS instance by using DMS.

Use the pgAdmin 4 client to connect to an ApsaraDB RDS instance

1. Add the IP address of the pgAdmin client to an IP address whitelist of the ApsaraDB RDS instance.

For more information about how to configure a whitelist, see Configure an IP address whitelist.

2. Start the pgAdmin 4 client.

> **Note** For information about how to download the pgAdmin 4 client, visit pgAdmin 4 (Windows).

3. Right-click **Servers** and choose **Create > Server**, as shown in the following figure.
4. On the **General** tab of the **Create - Server** dialog box, enter the name of the server, as shown in the following figure.
5. Click the **Connection** tab and enter the information of the instance, as shown in the following figure.

Parameter	Description
Host name/address	The internal endpoint of the ApsaraDB RDS instance. For more information about how to view the internal endpoint, see View and modify the internal endpoint and port number.
Port	The internal port number that is used to connect to the ApsaraDB RDS instance. For more information about how to view the internal port number, see View and modify the internal endpoint and port number.
Username	The name of the privileged account on the ApsaraDB RDS instance. For more information about how to obtain a privileged account, see Create a database and an account.
Password	The password of the privileged account of the ApsaraDB RDS instance.

6. Click **Save**.
7. If the connection information is correct, choose **Servers > Server Name > Databases > postgres**. If the following page appears, the connection is established.

> **Notice** The postgres database is the default system database of the ApsaraDB RDS instance. Do not perform operations on this database.

5.Instances

5.1. Create an instance

This topic describes how to create an ApsaraDB RDS for PostgreSQL instance in the ApsaraDB RDS console.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, click **Create Instance** in the upper-right corner.
3. Configure the parameters described in the following table.

Section	Parameter	Description
Basic Configurations	**Organization**	The organization to which the instance belongs.
	Resource Set	The resource set to which the instance belongs.
Area	**Region**	The region in which you want to create the instance. Services in different regions cannot communicate over an internal network. After the instance is created, the region cannot be changed.
	Primary Node Zone	The zone in which the primary instance is deployed.
	Deployment Method	Specifies whether to deploy the primary and secondary instances in separate zones. ApsaraDB RDS supports **Multi-zone Deployment** and **Single-zone Deployment**. If you select **Multi-zone Deployment**, you must configure **Secondary Node Zone**.
	Secondary Node Zone	The zone in which the secondary instance is deployed. This parameter is available only when you set the **Deployment Method** parameter to **Multi-zone Deployment**. **Note** If you select the same zone for both the primary and secondary instances, the deployment is equivalent to single-zone deployment.
	Database Engine	The database engine of the instance. Select **PostgreSQL**.

Section	Parameter	Description
Specifications	**Engine Version**	The version of the database engine. Valid values: ○ 10.0 ○ 11.0 ○ 12.0 ○ 13.0
	Edition	The edition of the instance. Select **High-availability Edition**.
	Storage Type	The storage type of the instance. Local SSDs and standard SSDs are supported. The storage types that are supported vary based on the engine version: ○ PostgreSQL 10.0: Local SSDs and standard SSDs are supported. ○ PostgreSQL 11.0: Standard SSDs are supported. ○ PostgreSQL 12.0: Local SSDs are supported. ○ PostgreSQL 13.0: Standard SSDs are supported.
	Encrypted	Specifies whether to encrypt standard SSDs. This parameter is available only when you set the **Storage Type** parameter to **Standard SSD**. If you select Encrypted, you must specify the **Encryption Key** parameter. If you do not have a key, you must first create one in the Key Management Service (KMS) console. For more information, see Create a CMK in *Key Management Service User Guide*.
	Encryption Key	The key that is used to encrypt standard SSDs. This parameter is available only when you select **Encrypted**.
	Instance Specifications	The specifications of the instance. The maximum number of connections and the maximum IOPS vary based on the memory size. The actual specifications are displayed in the console. For more information, see Instance types in *ApsaraDB RDS Product Introduction*.
	Storage Capacity	The storage capacity that is provided to store data files, system files, binlog files, and transaction files in the instance. Valid values: 20 to 6000. Unit: GB. The value must be in 1 GB increments.
	Connection Type	The connection type of the instance. Valid values: ○ **Internet**: Instances of this connection type can be connected over the Internet. ○ **Internal Network**: Instances of this connection type can be connected over an internal network. **Note** After the instance is created, the value of this parameter cannot be changed. Proceed with caution.

Section	Parameter	Description
Network	Network Type	The network type of the instance. Valid values: ○ **Classic Network**: Cloud services in the classic network are not isolated. Unauthorized access to a cloud service is blocked only by the security group or whitelist policy of the service. ○ **VPC**: A virtual private cloud (VPC) helps you build an isolated network environment on Apsara Stack. You can customize route tables, CIDR blocks, and gateways in a VPC. We recommend that you select VPC for improved security. Note ■ If you configure multi-zone deployment, you must create vSwitches for the zones of primary and secondary instances in the specified VPC. ■ If you select VPC, you must specify a VPC and a vSwitch.
	IP Whitelist	The IP address or CIDR block that is allowed to connect to the instance.
Instance Settings	Quantity	The number of instances that you want to create. Default value: 1.
	Instance Name	The name of the instance. ○ The name must be 2 to 64 characters in length. ○ The name must start with a letter. ○ The name can contain letters, digits, and the following special characters: _ - : ○ The name cannot start with http:// or https://.

4. Click **Submit**.

5.2. Create an ApsaraDB RDS for PostgreSQL instance that uses standard SSDs

Cloud disks are block-level data storage products provided by Alibaba Cloud for Elastic Compute Service (ECS). They provide low latency and high performance, durability, and reliability. This topic describes how to create one or more instances that use standard SSDs in the ApsaraDB RDS console.

Prerequisites

An instance that runs PostgreSQL 10.0 or later can be created.

Context

An ApsaraDB RDS instance with standard SSDs uses a distributed triplicate mechanism to ensure high data reliability. If service disruptions due to hardware failures occur within a zone, data within the zone is copied to an available disk in another zone to ensure data durability and availability.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, click **Create Instance** in the upper-right corner.
3. Configure the parameters described in the following table.

Section	Parameter	Description
Basic Configurations	**Organization**	The organization to which the instance belongs.
	Resource Set	The resource set to which the instance belongs.
Area	**Region**	The region in which you want to create the instance. Services in different regions cannot communicate over an internal network. After the instance is created, the region cannot be changed.
	Primary Node Zone	The zone in which the primary instance is deployed.
	Deployment Method	Specifies whether to deploy the primary and secondary instances in separate zones. ApsaraDB RDS supports **Multi-zone Deployment** and **Single-zone Deployment**. If you select **Multi-zone Deployment**, you must configure **Secondary Node Zone**.
	Quantity	The number of instances that you want to create. Default value: 1.
	Instance Name	The name of the instance. ◦ The name must be 2 to 64 characters in length. ◦ The name must start with a letter. ◦ The name can contain letters, digits, and the following special characters: _ - : ◦ The name cannot start with http:// or https://.

Section	Parameter	Description
Specifications	Connection Type	The connection type of the instance. Valid values: ○ **Internet**: Instances of this connection type can be connected over the Internet. ○ **Internal Network**: Instances of this connection type can be connected over an internal network. **Note** After the instance is created, the value of this parameter cannot be changed. Proceed with caution.
	Database Engine	The database engine of the instance. Select **PostgreSQL**.
	Engine Version	The version of the database engine. Set the value to **10.0** or a later version number.
	Edition	The edition of the instance. For more information, see Instance types in *ApsaraDB RDS Product Introduction*.
	Storage Type	The storage type of the instance. Select **Standard SSD**.
	Encrypted	Specifies whether to encrypt standard SSDs. If you select Encrypted, you must specify the **Encryption Key** parameter. If you do not have a key, you must first create one in the Key Management Service (KMS) console. For more information, see Configure data encryption. **Note** Disk encryption provides maximum protection for your data with minimal impact on your workloads or applications. In addition, ApsaraDB RDS automatically applies disk encryption to both the snapshots that are generated from the encrypted standard SSDs and to the standard SSDs that are created from those snapshots.
	Encryption Key	The key that is used to encrypt standard SSDs. This parameter is available only when you select **Encrypted**.
	Instance Specifications	The instance specifications of the instance. The maximum number of connections and the maximum IOPS vary based on the memory size. The actual specifications are displayed in the console. For more information, see Instance types in *ApsaraDB RDS Product Introduction*.
	Storage Capacity	The storage capacity that is provided to store data files, system files, binlog files, and transaction files in the instance. Valid values: 20 to 600. Unit: GB. The value must be in 1 GB increments.

Section	Parameter	Description
Network	Network Type	The network type of the instance. Valid values: ○ **Classic Network**: Cloud services in the classic network are not isolated. Unauthorized access to a cloud service is blocked only by the security group or whitelist policy of the service. ○ **VPC**: A virtual private cloud (VPC) helps you build an isolated network environment on Apsara Stack. You can customize route tables, CIDR blocks, and gateways in a VPC. We recommend that you select VPC for improved security. Note ■ If you configure multi-zone deployment, you must create vSwitches for the zones of primary and secondary instances in the specified VPC. ■ If you select VPC, you must specify a VPC and a vSwitch.
	IP Whitelist	The IP address or CIDR block that is allowed to connect to the instance.

4. Click **Submit**.

5.3. View basic information of an instance

This topic describes how to view the details of an ApsaraDB RDS instance, such as its basic information, internal network connection information, status, and configurations.

Procedure

1. Log on to the ApsaraDB RDS console.
2. Use one of the following methods to go to the **Basic Information** page of an instance:
   - On the **Instances** page, click the ID of the instance to go to the **Basic Information** page.
   - On the **Instances** page, find the instance and click **Manage** in the **Actions** column to go to the **Basic Information** page.

5.4. Restart an instance

This topic describes how to manually restart an ApsaraDB RDS instance. This applies if the number of connections exceeds the specified threshold or if an instance has performance issues.

Prerequisites

The instance is in the **Running** state.

Procedure

1. Log on to the ApsaraDB RDS console.

2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the upper-right corner of the page, click **Restart Instance**.

   > **Note** When you restart an instance, applications are disconnected from the instance. We recommend that you make appropriate service arrangements before you restart an instance. Proceed with caution.

5. In the message that appears, click **Confirm**.

5.5. Change the specifications of an instance

This topic describes how to change the specifications of an ApsaraDB RDS instance. You can upgrade or downgrade an ApsaraDB RDS instance to meet your business needs.

Prerequisites

The instance is in the **Running** state and is not in the backing up or restoring state.

Procedure

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the **Configure Information** section of the **Basic Information** page, click **Change Specifications**.
5. On the **Change Specifications** page, set **Edition**, **Instance Type**, and **Storage Capacity**.
6. Click **Submit**.

5.6. Set a maintenance window

This topic describes how to set a maintenance window for an ApsaraDB RDS instance.

Context

The backend system performs maintenance on the ApsaraDB RDS instances during the maintenance window. This ensures the stability of the ApsaraDB RDS instance. The default maintenance window is from 02:00 (UTC+8) to 06:00 (UTC+8). We recommend that you set the maintenance window to off-peak hours of your business to avoid impacts on your business.

Precautions

- An instance enters the **Maintaining Instance** state before the maintenance window to ensure stability during the maintenance process. When the instance is in this state, access to data in the database and query operations such as performance monitoring are not affected. However, except for account and database management and IP address whitelist configuration, modification

operations such as upgrade, downgrade, and restart are temporarily unavailable.

- During the maintenance window, one or two network interruptions may occur. Make sure that your applications are configured with automatic reconnection policies.

Procedure

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the **Configuration Information** section, click **Configure** to the right of **Maintenance Window**.
5. Select a maintenance window and click **Save**.

> **Note** The maintenance window is displayed in UTC+8.

5.7. Configure primary/secondary switchover

ApsaraDB RDS provides the primary/secondary switchover feature to ensure the high availability of databases. The primary/secondary switchover is performed when the primary instance becomes unavailable. You can also manually switch your business to the secondary instance. This topic describes how to manually switch over services between a primary instance and its secondary instance.

Context

Data is synchronized in real time between the primary and secondary instances. You can access only the primary instance. The secondary instance serves only as a backup instance and does not allow external access. After the switchover, the original primary instance becomes the secondary instance.

> **Note** Network interruptions may occur during a switchover. Make sure that your applications are configured with automatic reconnection policies.

Procedure

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Service Availability**.
5. In the **Availability Information** section, click **Switch Primary/Secondary Instance**.
6. In the **Switch Primary/Secondary Instance** message, click **OK**.

> Note
>
> - During the switchover, operations such as managing databases and accounts and changing network types cannot be performed. Therefore, we recommend that you select Switch Within Maintenance Window.
> - For more information about how to set a maintenance window, see Set a maintenance window.

5.8. Release an instance

This topic describes how to manually release an instance.

Precautions

- Only instances in the running state can be released.
- After an instance is released, the instance data is immediately deleted. We recommend that you back up your data before you release an instance.
- When you release a primary instance, all of its read-only instances are also released.

Procedure

1. Log on to the ApsaraDB RDS console.
2. Find the instance that you want to release and choose **More > Release Instance** in the Actions column.
3. In the **Release Instance** message, click **Confirm**.

5.9. Modify parameters of an instance

This topic describes how to view and modify the values of some parameters and query parameter modification records in the console.

Precautions

- To ensure instance stability, you can modify only specific parameters in the ApsaraDB RDS console.
- When you modify parameters on the **Editable Parameters** tab, refer to the **Value Range** column corresponding to each parameter.
- After specific parameters are modified, you must restart your instance for the changes to take effect. The necessity of restart is displayed in the **Force Restart** column on the **Editable Parameters** tab. We recommend that you modify the parameters of an instance during off-peak hours and make sure that your applications are configured with automatic reconnection policies.

Modify parameters

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Parameters**.

5. Perform the following operations:

   Export the parameter settings of the instance to your computer.

   On the Editable Parameters tab, click **Export Parameters**. The parameter settings of the instance are exported as a TXT file to your computer.

   Modify and import the parameter settings.

   i. After you modify parameters in the exported parameter file, click **Import Parameters** and copy the parameter settings to the field.
   ii. Click **OK**.
   iii. In the upper-right corner of the page, click **Apply Changes**.

   > **Note**
   > - If the new parameter value takes effect only after an instance restart, the system prompts you to restart the instance. We recommend that you restart the instance during off-peak hours and make sure that your applications are configured with automatic reconnection policies.
   > - Before the new parameter values are applied, you can click **Cancel Changes** to cancel them.

   Modify a single parameter.

   i. On the **Editable Parameters** tab, find the parameter that you want to modify and click the icon in the **Actual Value** column.
   ii. Enter a new value based on the prompted value range.
   iii. Click **Confirm**.
   iv. In the upper-right corner of the page, click **Apply Changes**.

   > **Note**
   > - If the new parameter value takes effect only after an instance restart, the system prompts you to restart the instance. We recommend that you restart the instance during off-peak hours and make sure that your applications are configured with automatic reconnection policies.
   > - Before the new parameter value is applied, you can click **Cancel Changes** to cancel it.

View the parameter modification history

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Parameters**.
5. On the page that appears, click the **Edit History** tab.
6. Select a time range and click **Search**.

5.10. Read-only instances

5.10.1. Overview of read-only ApsaraDB RDS for PostgreSQL instances

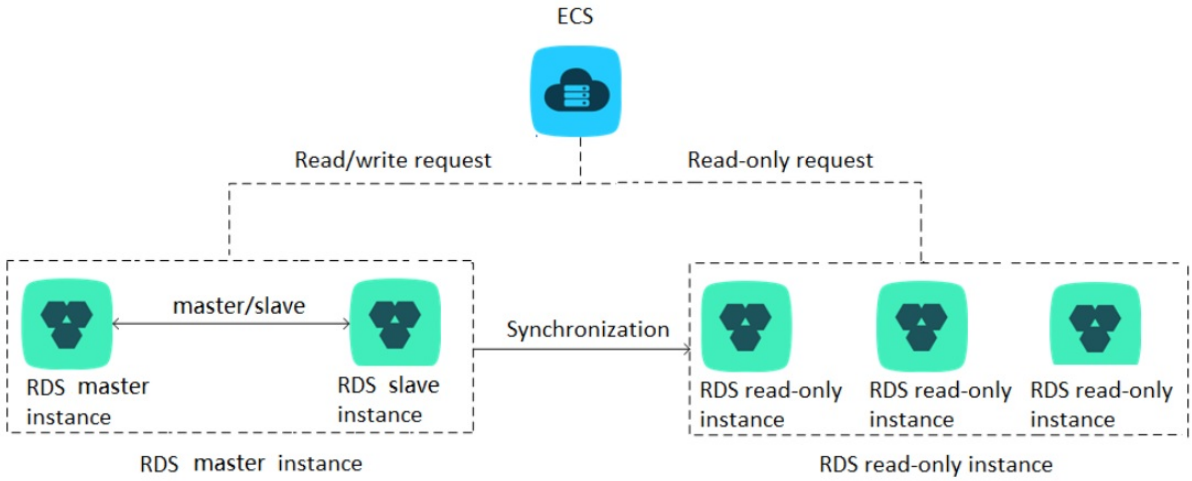
This topic provides an overview of read-only ApsaraDB RDS for PostgreSQL instances. If your database system receives a small number of write requests but a large number of read requests, a single primary RDS instance may be overwhelmed by read requests and have its workloads affected. To offload read requests from the primary RDS instance, you can create one or more read-only RDS instances. Read-only RDS instances help increase the read capability of your database system and the throughput of your application.

Overview

When a read-only RDS instance is being created, ApsaraDB RDS replicates data from the secondary RDS instance to the read-only RDS instance. After the read-only RDS instance is created, it has the same data as the primary RDS instance. In addition, after the data on the primary RDS instance is updated, ApsaraDB RDS immediately synchronizes the updates to all read-only RDS instances that are attached to the primary RDS instance.

> **Note**
>
> - RDS instances that run PostgreSQL 10.0, 11.0, 12.0, or 13.0 support read-only instances.
> - The primary RDS instance must use standard SSDs.
> - The primary RDS instance must have at least eight CPU cores and 32 GB of memory.
> - Each read-only RDS instance runs in a single-node architecture. In this architecture, no secondary RDS instance is provided as a standby for a read-only RDS instance.

The following figure shows the topology of the primary RDS instance and its read-only RDS instances.



Features

- Read-only RDS instances reside within the same region as the primary RDS instance, but can reside in different zones.

- The specifications and storage space of read-only RDS instances cannot be lower than those of the primary RDS instance.
- The network types of read-only RDS instances can differ from the network type of the primary RDS instance.
- The databases and accounts on read-only RDS instances are synchronized from the primary RDS instance. You do not need to manage databases or accounts on read-only RDS instances.
- When you create a read-only RDS instance, ApsaraDB RDS replicates the IP address whitelists of the primary RDS instance to the read-only RDS instance. However, the IP address whitelists of the read-only RDS instance are independent of the IP address whitelists of the primary RDS instance. For more information about how to modify the IP address whitelists of a read-only RDS instance, see Configure an IP address whitelist.
- Read-only RDS instances support monitoring and alerting. You can monitor metrics such as disk usage, IOPS, number of connections, and CPU utilization.

Limits

- A maximum of five read-only RDS instances can be created on a primary RDS instance.
- You cannot configure backup policies or manually create backups for read-only RDS instances. These operations are performed on primary RDS instances.
- You cannot migrate data to read-only RDS instances.
- You cannot create or delete databases on read-only RDS instances.
- You cannot create or delete accounts, grant permissions to accounts, or change the passwords of accounts on read-only RDS instances.

FAQ

Q: After I create accounts on my primary RDS instance, can I manage the accounts on the read-only RDS instances of my primary RDS instance?

No, you cannot manage the accounts on the read-only RDS instances. Although the accounts created on your primary RDS instance are synchronized to the read-only RDS instances, the accounts have only the read permissions on the read-only RDS instances.

5.10.2. Create a read-only ApsaraDB RDS for PostgreSQL instance

This topic describes how to create a read-only instance for your primary ApsaraDB RDS for PostgreSQL instance. This allows your database system to process a large number of read requests and increases the throughput of your application. The data on each read-only instance is a copy of that of the primary instance. Data updates to the primary instance are synchronized to all of its read-only instances.

Prerequisites

- The primary instance runs PostgreSQL 10.0.
- The specifications of the primary instance must have at least eight CPU cores and 32 GB of memory.

Precautions

- You can create read-only instances only for your primary instance. You cannot change existing

instances to read-only instances.

- When you create a read-only instance, the system replicates data from a secondary instance. Therefore, operations on your primary instance are not interrupted.
- A read-only instance does not inherit the parameter settings of the primary instance. The system generates default parameter settings for the read-only instance, and you can modify the settings in the ApsaraDB RDS console.
- The instance type and storage capacity of a read-only instance cannot be lower than that of the primary instance.
- You can create up to five read-only instances.

Create a read-only instance

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the Distributed by Instance Role section of the Basic Information page, click **Create Read-only Instance**.
5. On the Create Read-only Instance page, configure parameters and click **Submit**. The following table describes the parameters.

Section	Parameter	Description
Region	Region	The region where the instance is deployed.
Specifications	Database Engine	The database engine of the read-only instance, which is the same as that of the primary instance and cannot be changed.
	Engine Version	The engine version of the read-only instance, which is the same as that of the primary instance and cannot be changed.
	Edition	The edition of the read-only instance, which is the same as that of the primary instance and cannot be changed.
	Instance Type	The instance type of the read-only instance. The instance type of the read-only instance can be different from that of the primary instance, and can be changed at any time to facilitate flexible upgrade and downgrade. To ensure sufficient I/O throughput for data synchronization, we recommend that you select at least the same instance type as the primary instance for the read-only instance.

Section	Parameter	Description
	Storage Capacity	The storage capacity of the read-only instance. To ensure sufficient I/O throughput for data synchronization, we recommend that you select at least the same storage capacity as the primary instance for the read-only instance.
Network Type	Network Type	The network type of the read-only instance, which is the same as that of the primary instance and cannot be changed.
	VPC	Select a VPC if the network type is set to VPC.
	vSwitch	Select a vSwitch if the network type is set to VPC.

5.10.3. View a read-only ApsaraDB RDS for PostgreSQL instance

This topic describes how to view details of a read-only ApsaraDB RDS for PostgreSQL instance. You can go to the Basic Information page of a read-only instance from the Instances page or the read-only instance list of the primary instance. Read-only instances are managed in the same manner as primary instances. The Basic Information page shows the management operations that can be performed.

View instance details of a read-only instance by using its ID

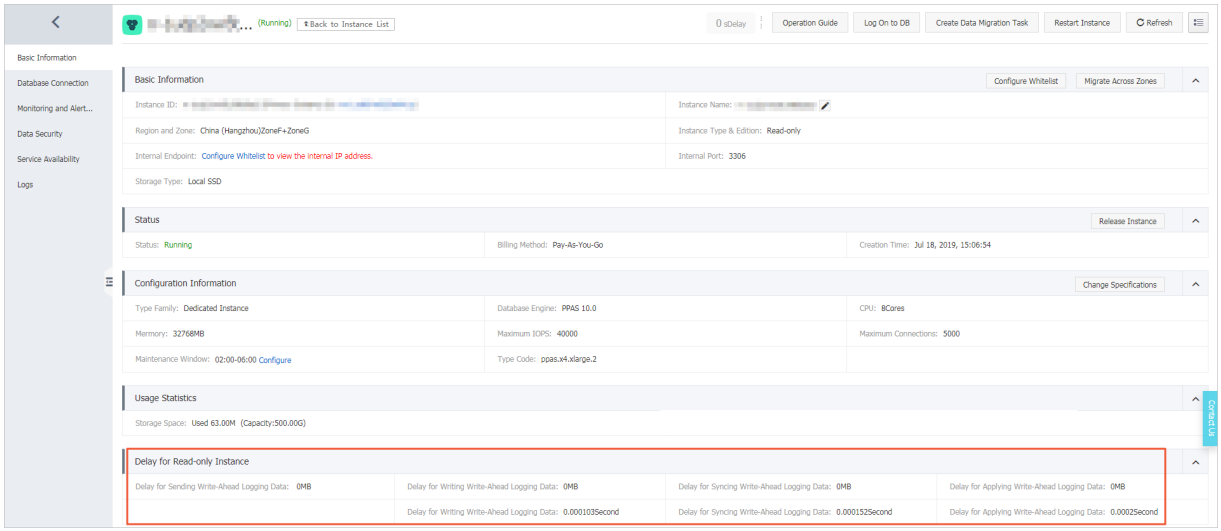
1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the instance that you want to view.
3. Click the ID of the instance or click **Manage** in the corresponding Actions column to go to the Basic Information page.

View details of a read-only instance by using the primary instance

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. On the **Basic Information** page, move the pointer over the number below **Read-only Instance** in the **Distributed by Instance Role** section. The ID of the read-only instance is displayed.
5. Click the ID of the read-only instance to go to the Basic Information page of the read-only instance.

View the latency of a read-only instance

When a read-only instance synchronizes data from its primary RDS instance, latency may occur. You can navigate to the Basic Information page of the read-only instance to view the latency of data synchronization to the instance.



5.10.4. Manage a read-only ApsaraDB RDS for PostgreSQL instance

This topic describes how to change specifications of a read-only RDS instance, release a read-only RDS instance, and view monitoring and alerting information of a read-only RDS instance.

Change specifications of a read-only RDS instance

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the instance that you want to manage.
3. Click the ID of the instance or click **Manage** in the Actions column to go to the Basic Information page.
4. In the **Configuration Information** section of the **Basic Information** page, click **Change Specifications**.
5. On the **Change Specifications** page, configure the parameters described in the following table and click **Submit**.

Parameter	Description
Edition	The edition of the read-only instance. It cannot be modified.
Storage Type	The storage type of the read-only instance. It is automatically set to Standard SSD and cannot be modified.
Instance Type	The instance type of the read-only instance. For more information, see Instance types in *ApsaraDB RDS Production Introduction*. >
Storage Capacity	The storage capacity that is provided to store data files, system files, binlog files, and transaction files in the instance. Valid values: 20 to 6000. Unit: GB. This value must be in 1 GB increments.

Release a read-only RDS instance

1. Log on to the ApsaraDB RDS console.

2. On the **Instances** page, find the instance that you want to manage.
3. Click the ID of the instance or click **Manage** in the Actions column to go to the Basic Information page.
4. In the **Status** section of the **Basic Information** page, click **Release Instance**. In the message that appears, click **Confirm**.

View monitoring and alerting information of a read-only RDS instance

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the instance that you want to manage.
3. Click the ID of the instance or click **Manage** in the Actions column to go to the Basic Information page.
4. In the left-side navigation pane, click **Monitoring and Alerts**.

> **Note** The monitoring and alerting feature of read-only RDS instances can be used in the same manner as that of the primary instance. For more information, see View monitored resources.

6.Database connection

6.1. Connect to an ApsaraDB RDS for PostgreSQL instance

This topic describes how to use Data Management (DMS) or the pgAdmin 4 client to connect to an ApsaraDB RDS instance.

Context

You can log on to DMS from the ApsaraDB RDS console and then connect to an ApsaraDB RDS instance.

DMS is a data management service that integrates data, schema, and server management, access security, BI charts, data trends, data tracking, and performance optimization. DMS can be used to manage relational and non-relational databases, such as MySQL, SQL Server, PostgreSQL, MongoDB, and Redis. It can also be used to manage Linux servers.

You can also use a client to connect to an ApsaraDB RDS instance. ApsaraDB RDS for PostgreSQL is fully compatible with PostgreSQL. You can connect to an ApsaraDB RDS for PostgreSQL instance in a similar manner as you would connect to an open source PostgreSQL instance. In this topic, the pgAdmin 4 client is used to connect to an ApsaraDB RDS instance.

Use DMS to connect to an ApsaraDB RDS instance

For more information about how to use DMS to connect to an ApsaraDB RDS instance, see Log on to an ApsaraDB for RDS instance by using DMS.

Use the pgAdmin 4 client to connect to an ApsaraDB RDS instance

1. Add the IP address of the pgAdmin client to an IP address whitelist of the ApsaraDB RDS instance. For more information about how to configure a whitelist, see Configure an IP address whitelist.
2. Start the pgAdmin 4 client.

   > **Note** For information about how to download the pgAdmin 4 client, visit pgAdmin 4 (Windows).

3. Right-click **Servers** and choose **Create > Server**, as shown in the following figure.
4. On the **General** tab of the **Create - Server** dialog box, enter the name of the server, as shown in the following figure.
5. Click the **Connection** tab and enter the information of the instance, as shown in the following figure.

Parameter	Description
Host name/address	The internal endpoint of the ApsaraDB RDS instance. For more information about how to view the internal endpoint, see View and modify the internal endpoint and port number.

Parameter	Description
Port	The internal port number that is used to connect to the ApsaraDB RDS instance. For more information about how to view the internal port number, see View and modify the internal endpoint and port number.
Username	The name of the privileged account on the ApsaraDB RDS instance. For more information about how to obtain a privileged account, see Create a database and an account.
Password	The password of the privileged account of the ApsaraDB RDS instance.

6. Click **Save**.
7. If the connection information is correct, choose **Servers > Server Name > Databases > postgres**. If the following page appears, the connection is established.

> **Notice** The postgres database is the default system database of the ApsaraDB RDS instance. Do not perform operations on this database.

6.2. Use DMS to log on to an ApsaraDB RDS instance

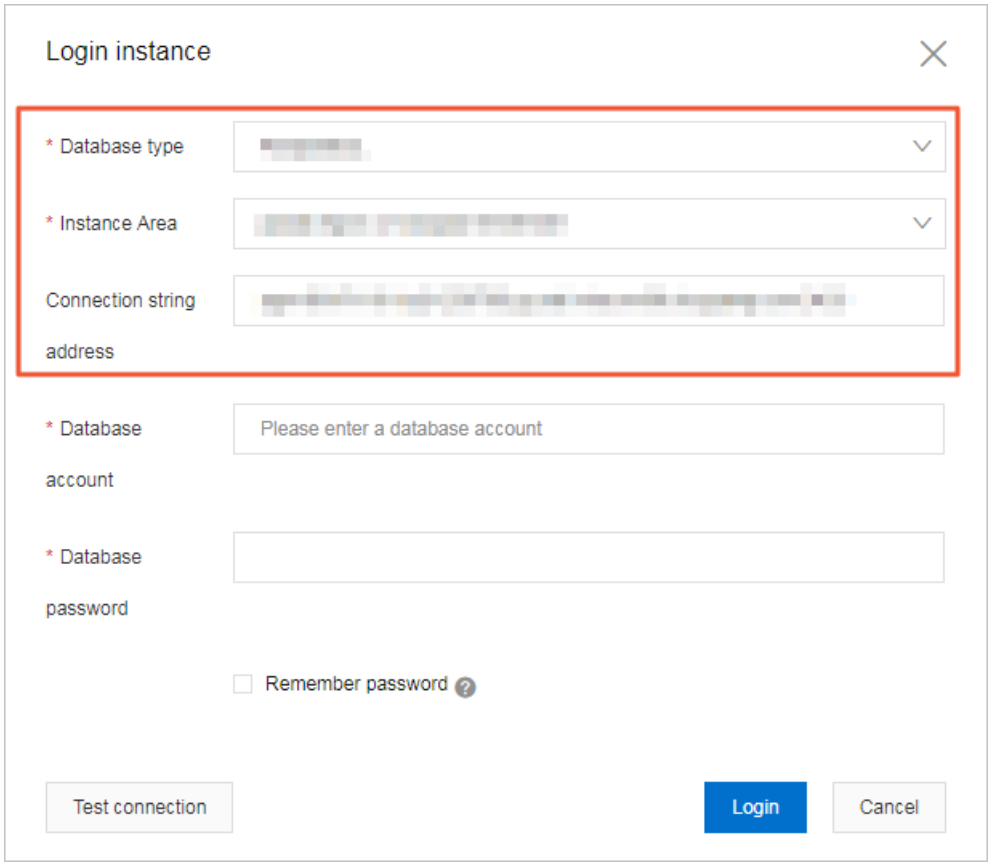
This topic describes how to use Data Management (DMS) to log on to an ApsaraDB RDS instance.

Prerequisites

The IP address whitelist is configured. For more information about how to configure an IP address whitelist, see Configure an IP address whitelist.

Procedure

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. Click **Log On to DB** in the upper-right corner of the page.
5. In the **Login instance** dialog box of the **DMS** console, check values of **Database type**, **Instance Area**, and **Connection string address**. If the information is correct, enter **Database account** and **Database password**, as shown in the following figure.



Parameter	Description
Database type	The engine of the database. By default, the engine of the database to be connected is displayed.
Instance Area	The region where the instance is deployed. By default, the region of the current instance is displayed.
Connection string address	The endpoint of the instance. By default, the endpoint of the current instance is displayed.
Database account	The account of the database to be connected.
Database password	The password of the account used to connect to the database.

6. Click **Login**.

> **Note** If you want the browser to remember the password, select **Remember password** before you click **Login**.

6.3. View and modify the internal endpoint and port number

You must use the internal endpoint and port number to access an ApsaraDB RDS instance. This topic describes how to view and modify the internal endpoint and port number of an ApsaraDB RDS instance in the ApsaraDB RDS console.

View the internal endpoint and port number

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the **Basic Information** section, view the internal endpoint and port number of the instance.

Modify the internal endpoint and port number

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Database Connection**.
5. On the right side of the page, click **Change Endpoint**.
6. In the dialog box that appears, set **Connection Type** to **Internal Endpoint**.
7. Modify the endpoint prefix and port number and click **OK**.

FAQ

- Q: Do I need to modify the endpoint or port number in my application after I modify the endpoint or port number of an instance?

  A: Yes, you must modify the endpoint or port number in the application after you modify them. Otherwise, the application cannot connect to databases of the instance.

- Q: Does the modification of the endpoint take effect immediately? Do I need to restart the instance?

  A: No, you do not need to restart the instance. The modification takes effect immediately.

7.Accounts

7.1. Create an account

Before you start to use ApsaraDB RDS, you must create an account on an ApsaraDB RDS instance. This topic describes how to create an account on an ApsaraDB RDS for PostgreSQL instance.

Account types

ApsaraDB RDS for PostgreSQL instances support two types of accounts: privileged accounts and standard accounts. The following table describes these account types.

Account type	Description
Privileged account	• You can create and manage privileged accounts only by using the ApsaraDB RDS console or the API. • If your ApsaraDB RDS instance uses local SSDs, you can create only a single privileged account. If your ApsaraDB RDS instance uses standard or enhanced SSDs, you can create more than one privileged account. A privileged account allows you to manage all the standard accounts and databases that are created on your ApsaraDB RDS instance. • A privileged account has more permissions that allow you to manage your ApsaraDB RDS instance at more fine-grained levels. For example, you can grant the query permissions on different tables to different users. • A privileged account has the permissions to disconnect accounts that are created on your ApsaraDB RDS instance.
Standard account	• You can create and manage standard accounts by using the ApsaraDB RDS console, API operations, or SQL statements. • You can create more than one standard account on your ApsaraDB RDS instance. • You must grant the permissions on specific databases to a standard account. • A standard account does not have the permissions to create, manage, or disconnect other accounts on your ApsaraDB RDS instance.

Precautions

- If your ApsaraDB RDS instance uses local SSDs, you can create one privileged account in the ApsaraDB RDS console. After the privileged account is created, it cannot be deleted. You can also create and manage more than one standard account by using SQL statements.
- If your ApsaraDB RDS instance uses standard or enhanced SSDs, you can create more than one privileged account and standard account in the ApsaraDB RDS console. You can also create and manage more than one standard account by using SQL statements.
- To migrate data from an on-premises database to your ApsaraDB RDS instance, you must create a database and an account on the ApsaraDB RDS instance. Make sure that the created database has the same properties as the on-premises database. Also make sure that the created account has the same permissions on the created database as the account that is authorized to manage the on-premises database.
- Follow the least privilege principle to create accounts and grant them read-only permissions or read

and write permissions on databases. If necessary, you can create more than one account and grant them only the permissions on specific databases. If an account does not need to write data to a database, grant only the read-only permissions on that database to the account.

- For security purposes, we recommend that you specify strong passwords for the accounts on your ApsaraDB RDS instance and change the passwords on a regular basis.

Create a privileged account on an instance that uses local SSDs

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Accounts**.
5. On the Accounts page, click **Create Privileged Account** and configure the following parameters.

Parameter	Description
Database Account	○ The name of the account must be 2 to 16 characters in length. ○ The name can contain lowercase letters, digits, and underscores (_). ○ The name must start with a letter and end with a letter or digit.
Password	○ The password of the account must be 8 to 32 characters in length. ○ The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters.characters. ○ Special characters include ! @#$%^&*()_+-=
Re-enter Password	Enter the password of the account again.

6. Click **Create**.

Create a privileged or standard account on an instance that uses standard or enhanced SSDs

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Accounts**.
5. On the Accounts page, click **Create Account** and configure the following parameters.

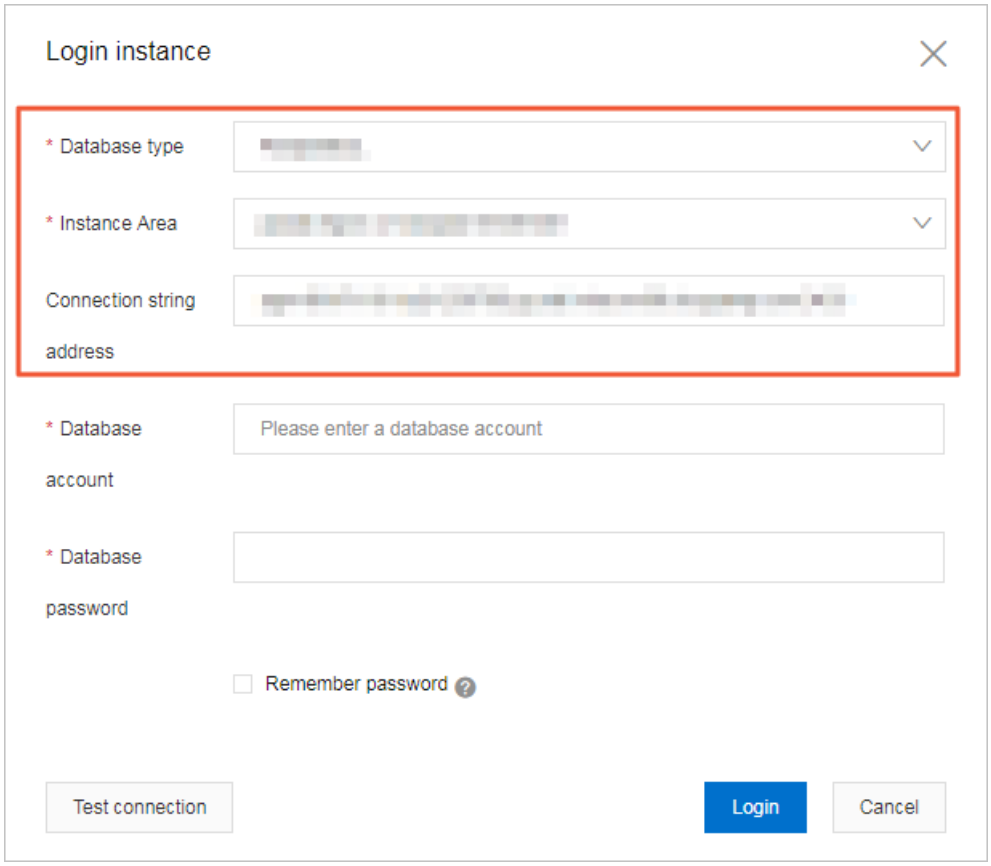
Parameter	Description

Parameter	Description
Database Account	○ The name of the account must be 2 to 16 characters in length. ○ The name can contain lowercase letters, digits, and underscores (_). ○ The name must start with a letter and end with a letter or digit.
Account Type	Select **Privileged Account** or **Standard Account**.
Password	○ The password of the account must be 8 to 32 characters in length. ○ The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters.characters. ○ Special characters include ! @#$%^&*()_+-=
Re-enter Password	Enter the password of the account again.
Description	This parameter is optional. You can enter relevant description to make the instance identifiable. The description can be up to 256 characters in length.

6. Click **Create**.

Create a standard account on an instance that uses local SSDs

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. Click **Log On to DB** in the upper-right corner of the page.
5. In the **Login instance** dialog box of the **DMS** console, check values of **Database type**, **Instance Area**, and **Connection string address**. If the information is correct, enter **Database account** and **Database password**, as shown in the following figure.



Parameter	Description
Database type	The engine of the database. By default, the engine of the database to be connected is displayed.
Instance Area	The region where the instance is deployed. By default, the region of the current instance is displayed.
Connection string address	The endpoint of the instance. By default, the endpoint of the current instance is displayed.
Database account	The account of the database to be connected.
Database password	The password of the account used to connect to the database.

6. Click **Login**. If you want the browser to remember the password, select **Remember password** before you click **Login**.

   > **Note** If a message prompts you that the connection fails, the problem may be caused by an improperly configured whitelist. Reconfigure the whitelist in the console. For more information, see Configure an IP address whitelist.

7. The **SQLConsole** page appears after you log on to the instance. Execute a statement in the following format to create a standard account:

```
CREATE USER name [ [ WITH ] option [ ... ] ]
where option can be:
    SUPERUSER | NOSUPERUSER
  | CREATEDB | NOCREATEDB
  | CREATEROLE | NOCREATEROLE
  | CREATEUSER | NOCREATEUSER
  | INHERIT | NOINHERIT
  | LOGIN | NOLOGIN
  | REPLICATION | NOREPLICATION
  | CONNECTION LIMIT connlimit
  | [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
  | VALID UNTIL 'timestamp'
  | IN ROLE role_name [, ...]
  | IN GROUP role_name [, ...]
  | ROLE role_name [, ...]
  | ADMIN role_name [, ...]
  | USER role_name [, ...]
  | SYSID uid
```

For example, if you want to create a user account named test2 whose password is 123456, execute the following statement:

```
create user test2 password '123456';
```

8. Click **execute**.

7.2. Reset the password

This topic describes how to use the ApsaraDB RDS console to reset the password of your database account if you forget the password.

Procedure

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Accounts**.
5. In the **Actions** column corresponding to the account, click **Reset Password**.
6. In the dialog box that appears, enter a new password and click **OK**.

> **Note** The password must meet the following requirements:
>
> - The password must be 8 to 32 characters in length.
> - The password must contain at least three of the following characters: uppercase letters, lowercase letters, digits, and special characters.
> - Special characters include ! @ # $ % ^ & * ( ) _ + - =

7.3. Lock an account

You can lock a database account in the ApsaraDB RDS console to make the account unavailable.

Procedure

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Accounts**.
5. Find the account that you want to lock and click **Lock** in the **Actions** column.
6. To unlock the account, click **Unlock** in the **Actions** column.

7.4. Delete an account

You can delete a database account in the ApsaraDB RDS console.

Prerequisites

You can use the console to delete privileged and standard accounts that are no longer used.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the instance that you want to manage.
3. In the left-side navigation pane, click **Accounts**.
4. Find the account that you want to delete and click **Delete** in the Actions column.
5. In the message that appears, click **Confirm**.

> **Note** Accounts in the **Processing** state cannot be deleted.

8.Databases

8.1. Create a database

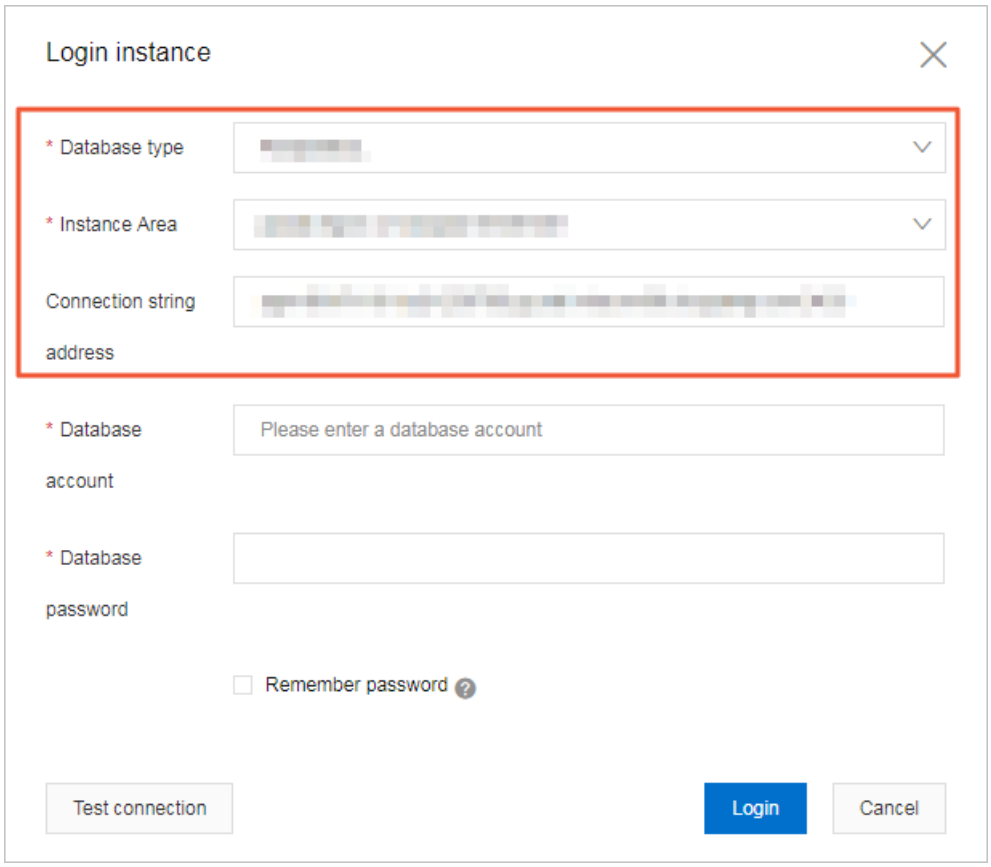
Before you start to use ApsaraDB RDS, you must create a database on an ApsaraDB RDS instance. This topic describes how to create a database on an ApsaraDB RDS for PostgreSQL instance.

Prerequisites

- An ApsaraDB RDS for PostgreSQL instance is created. For more information, see Create an instance.
- An account is created. For more information, see Create an account.

Procedure

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. Click **Log On to DB** in the upper-right corner of the page.
5. In the **Login instance** dialog box of the **DMS** console, check values of **Database type**, **Instance Area**, and **Connection string address**. If the information is correct, enter **Database account** and **Database password**, as shown in the following figure.



Parameter	Description
Database type	The engine of the database. By default, the engine of the database to be connected is displayed.
Instance Area	The region where the instance is deployed. By default, the region of the current instance is displayed.
Connection string address	The endpoint of the instance. By default, the endpoint of the current instance is displayed.
Database account	The account of the database to be connected.
Database password	The password of the account used to connect to the database.

6. Click **Login**. If you want the browser to remember the password, select **Remember password** before you click **Login**.

   > **Note** If a message prompts you that the connection fails, the problem may be caused by an improperly configured whitelist. Reconfigure the whitelist in the console. For more information, see Configure an IP address whitelist.

7. The **SQLConsole** page appears after you log on to the instance. Execute a statement in the following format to create a database:

   ```
   CREATE DATABASE name;
   ```

   For example, if you want to create a database named test, execute the following statement:

   ```
   create database test;
   ```

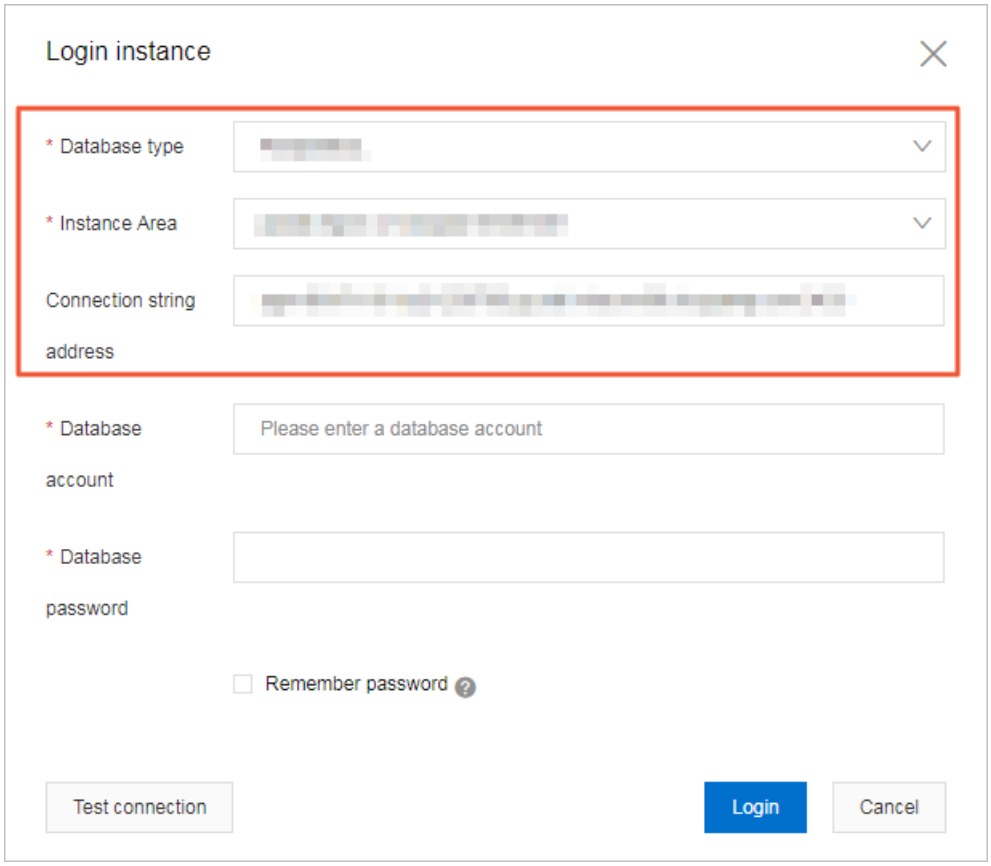
8. Click **execute**.

8.2. Delete a database

This topic describes how to delete a database in the ApsaraDB RDS for PostgreSQL console.

Procedure

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. Click **Log On to DB** in the upper-right corner of the page.
5. In the **Login instance** dialog box of the **DMS** console, check values of **Database type**, **Instance Area**, and **Connection string address**. If the information is correct, enter **Database account** and **Database password**, as shown in the following figure.



Parameter	Description
Database type	The engine of the database. By default, the engine of the database to be connected is displayed.
Instance Area	The region where the instance is deployed. By default, the region of the current instance is displayed.
Connection string address	The endpoint of the instance. By default, the endpoint of the current instance is displayed.
Database account	The account of the database to be connected.
Database password	The password of the account used to connect to the database.

6. Click **Login**. If you want the browser to remember the password, select **Remember password** before you click **Login**.

   > **Note** If a message prompts you that the connection fails, the problem may be caused by an improperly configured whitelist. Reconfigure the whitelist in the console. For more information, see Configure an IP address whitelist.

7. The **SQLConsole** page appears after you log on to the instance. Execute the following statement to delete a database:

```
drop database <database name>;
```

For example, if you want to delete a database named test2, execute the following statement:

```
drop database test2;
```

8. Click **execute**.

9.Networks, VPCs, and vSwitches

9.1. Change the VPC and vSwitch for an ApsaraDB RDS for PostgreSQL instance

This topic describes how to change the virtual private cloud (VPC) and vSwitch for an ApsaraDB RDS for PostgreSQL instance.

Prerequisites

The ApsaraDB RDS for PostgreSQL instance resides in a VPC.

Procedure

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Database Connection**.
5. In the upper-right corner of the Database Connection section, click **Switch vSwitch**.
6. Select a VPC and a vSwitch, and then click **OK**.

   > **Note** If you want to create a VPC and a vSwitch, you can click **go to the VPC console**.

7. In the message that appears, click **Switch**.

   > **Note**
   > - You may encounter a network interruption of about 30 seconds during the change process. Make sure that your application is configured to automatically reconnect to the instance.
   > - We recommend that you clear the cache immediately after the VPC and vSwitch are changed. Otherwise, data can be read but not written.

9.2. Change the network type of an ApsaraDB RDS for PostgreSQL instance

This topic describes how to change the network type of an ApsaraDB RDS for PostgreSQL instance between classic network and Virtual Private Cloud (VPC).

Prerequisites

The ApsaraDB RDS instance uses local SSDs.

Context

- Classic network: ApsaraDB RDS instances in the classic network are not isolated. You can block unauthorized access only by configuring IP address whitelists on these instances.
- VPC: Each VPC is an isolated network. We recommend that you use the VPC network type because it provides a higher security level.

  You can configure route tables, CIDR blocks, and gateways within a VPC. To smoothly migrate applications to the cloud, you can use the leased line or VPN method to create a virtual data center that consists of your self-managed data center and a VPC.

Change the network type from VPC to classic network

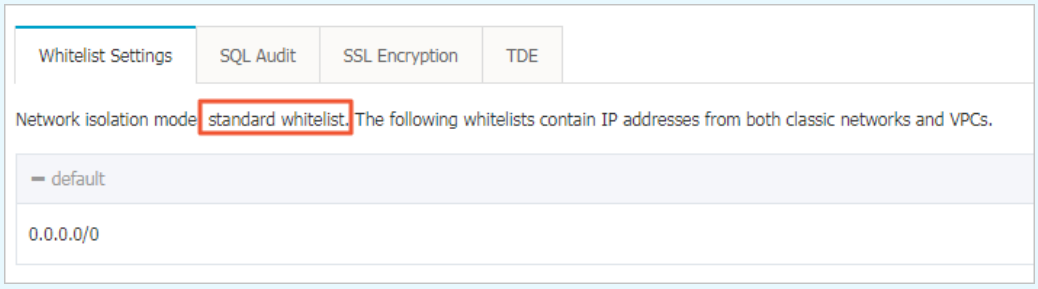
Precautions

- The ApsaraDB RDS instance must be in a VPC.
- After you change the network type from VPC to classic network, the internal endpoint of the ApsaraDB RDS instance remains unchanged. However, the IP address that is associated with the internal endpoint changes.
- After you change the network type from VPC to classic network, you cannot connect Elastic Compute Service (ECS) instances deployed in VPCs to the ApsaraDB RDS instance by using the internal endpoint. You must update the endpoint for the applications deployed on the ECS instances.
- You may encounter a network interruption of about 30 seconds during the change process. To avoid business interruptions, we recommend that you change the network type during off-peak hours or make sure that your application is configured to automatically reconnect to the instance.

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Database Connection**.
5. In the upper-right corner of the Database Connection section, click **Switch to Classic Network**.
6. In the message that appears, click **OK**.

   > **Note** After the network type is changed to classic network, only ECS instances within the classic network can connect to the ApsaraDB RDS instance by using the internal endpoint. You must configure the internal endpoint for the ECS instances.

7. Configure an IP address whitelist to allow ECS instances within the classic network to connect to the ApsaraDB RDS instance by using the internal endpoint.

> **Note**
>
> - If the network isolation mode of the ApsaraDB RDS instance is standard whitelist, add the internal IP addresses of the ECS instances to a whitelist of your ApsaraDB RDS instance.
>
> 
>
>
> - If the network isolation mode of the ApsaraDB RDS instance is enhanced whitelist, add the internal IP addresses of the ECS instances to a whitelist of the classic network type. If no whitelists of the classic network type are available, create a whitelist. For more information about the enhanced whitelist mode, see Switch to the enhanced whitelist mode.

Change the network type from classic network to VPC

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Database Connection**.
5. In the upper-right corner of the Database Connection section, click **Switch to VPC**.
6. In the Switch to VPC dialog box, select a VPC and a vSwitch and specify whether to retain the classic network endpoint.

> **Note**
>
> - Select a VPC. We recommend that you select the VPC where your ECS instances are deployed. Otherwise, the ECS instances cannot communicate with the ApsaraDB RDS instance over the internal network.
> - Select a vSwitch. If no vSwitches are available in the selected VPC, create one in the same zone where the ApsaraDB RDS instance is deployed. For more information, see Create a vSwitch in *Virtual Private Cloud User Guide*.
> - Clear or select the **Reserve Original Classic Network Endpoint** check box.
>
> - **Clear the Reserve Original Classic Network Endpoint check box**
>
> The classic network endpoint is not retained and changes to a VPC endpoint.
>
> When you change the network type from classic network to VPC, a network interruption of 30 seconds occurs. In this case, ECS instances located in the classic network are disconnected from your ApsaraDB RDS instance.
>
> - **Select the Reserve Original Classic Network Endpoint check box**
>
> The classic network endpoint is retained, and a new VPC endpoint is generated. In this case, your ApsaraDB RDS instance runs in hybrid access mode. Both ECS instances located in the classic network and ECS instances located in the selected VPC can access your ApsaraDB RDS instance over an internal network. You must set **Expiration Time (Important)** to **14 Days Later**, **30 Days Later**, **60 Days Later**, or **120 Days Later** for the classic network. You can also modify the expiration date after the network type is changed. For more information, see Configure hybrid access from both the classic network and VPCs.
>
> When you change the network type from classic network to VPC, no network interruptions occur. ECS instances located in the classic network are still connected with your ApsaraDB RDS instance until the classic network endpoint expires.
>
> Before the classic network endpoint expires, you must add the new VPC endpoint to your applications that run on the ECS instances located in the selected VPC. This allows ApsaraDB RDS to migrate your workloads to the selected VPC without network interruptions. Seven days before the classic network endpoint expires, the system sends a text message to the phone number bound to your Apsara Stack tenant account every day.
>
> For more information, see Hybrid access from both the classic network and VPCs.

7. Add the internal IP addresses of ECS instances located in the selected VPC to an IP address whitelist of the VPC network type. This allows the ECS instances to connect to your ApsaraDB RDS instance over an internal network. If no IP address whitelists of the VPC network type are available, create one.

> Note
>
> - If you retain the classic network endpoint, add the VPC endpoint to the ECS instances before the classic network endpoint expires.
> - If you do not retain the classic network endpoint, connections between ECS instances in the classic network and the ApsaraDB RDS instance over the internal network are interrupted. You must add the VPC endpoint to ECS instances in the VPC immediately after the network type is changed.

9.3. Configure hybrid access from both the classic network and VPCs

This topic describes how to use the hybrid access solution of ApsaraDB RDS to change the network type of an instance from classic network to Virtual Private Cloud (VPC) without network interruptions.

Prerequisites

- The ApsaraDB RDS instance uses local SSDs.
- The ApsaraDB RDS instance is deployed in the classic network.
- Available VPCs and vSwitches exist in the zone where the ApsaraDB RDS instance is deployed.

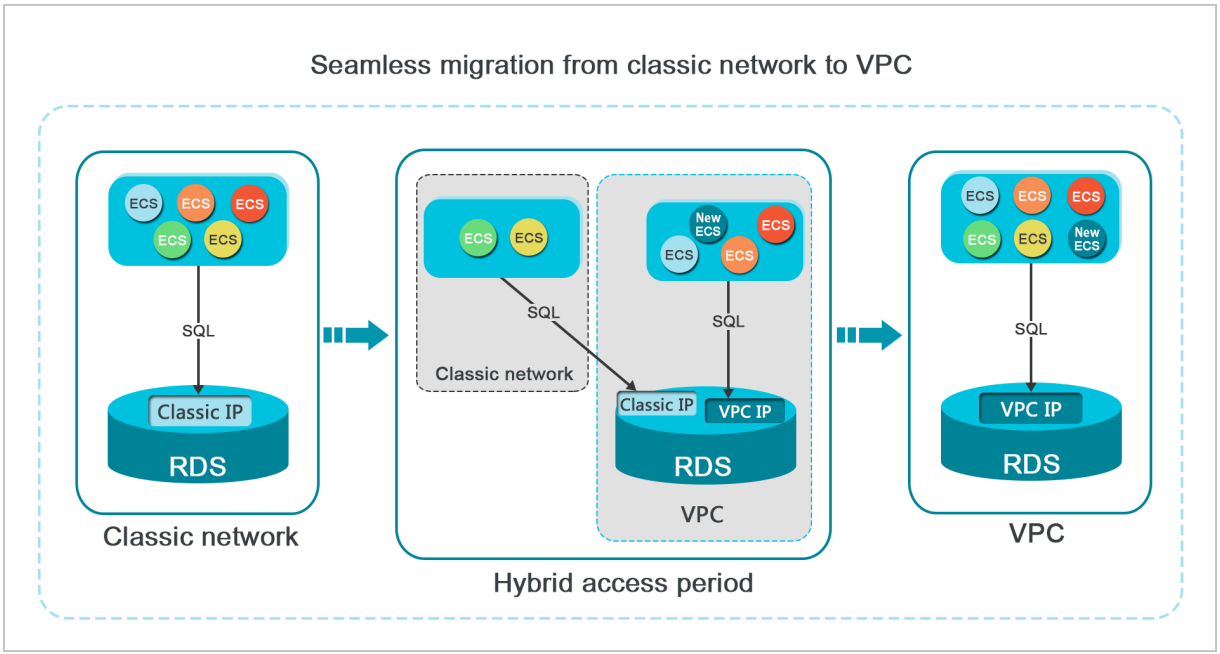
Context

In the past, when you change the network type of an ApsaraDB RDS instance from classic network to VPC, the internal endpoint of the ApsaraDB RDS instance would remain the same but the IP address bound to the endpoint would change to the corresponding IP address in the VPC. This change would cause a 30-second network interruption, and ECS instances within the classic network would not be able to access the ApsaraDB RDS instance by using the internal endpoint within this period. To smoothly change the network type, ApsaraDB RDS provides the hybrid access solution.

Hybrid access refers to the ability of an ApsaraDB RDS instance to be accessed by ECS instances in both the classic network and VPCs. During the hybrid access period, the ApsaraDB RDS instance reserves the original internal endpoint of the classic network and adds the internal endpoint of VPCs. This prevents network interruptions during the network type switchover.

For better security and performance, we recommend that you use the internal endpoint of VPCs. Hybrid access is available for a limited period of time. The internal endpoint of the classic network is released when the hybrid access period expires. In that case, your applications cannot access the ApsaraDB RDS database by using the internal endpoint of the classic network. You must configure the internal endpoint of VPCs in all your applications during the hybrid access period. This ensures smooth network switchover and minimizes the impact on your services.

For example, your company wants to use the hybrid access solution to change the network type from classic network to VPC. During the hybrid access period, some applications can access the database by using the internal endpoint of VPCs, and the other applications can access the database by using the original internal endpoint of the classic network. When all the applications access the database by using the internal endpoint of VPCs, the internal endpoint of the classic network can be released. The following figure illustrates the scenario.



Limits

During the hybrid access period, the instance has the following limits:

- The network type of your instance cannot be changed to classic network.
- Your instance cannot be migrated to another zone.

Change the network type from classic network to VPC

For more information, see Change the network type from classic network to VPC.

Change the expiration time for the original internal endpoint of the classic network

During the period in which your instance can be accessed over the classic network or VPCs, you can specify the expiration time for the endpoint of the classic network. The setting takes effect immediately. For example, if the endpoint of the classic network is about to expire on August 18, 2017 and you change the expiration time to 14 days later on August 15, 2017, the endpoint of the classic network is released on August 29, 2017.

To change the expiration time, perform the following steps:

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Database Connection**.

5. On the **Instance Connection** tab, click **Change Expiration Time**.
6. In the **Change Expiration Time** dialog box, select an expiration time and click **OK**.

10.Monitoring

10.1. View monitored resources

ApsaraDB RDS provides a wide range of performance metrics. This topic describes how to view resource monitoring data in the ApsaraDB RDS console.

Procedure

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Monitoring and Alerts**.
5. On the **Monitoring** tab, select a time range to query the corresponding monitoring data. The following table lists the specific metrics.

Metric	Description
CPU Utilization	The CPU utilization of the instance. Unit: %.
Memory Usage	The memory usage of the instance. Unit: %.
IOPS	The number of input and output operations that are performed per second.
Disk Space Used	The used disk space of the instance. Unit: MB.
Received Traffic	The inbound and outbound bandwidths of the instance.
Data Disk Usage	The data disk usage of the instance. Unit: %.

> **Note** You can click **Refresh** in the upper-right corner of the **Monitoring** tab to refresh the monitoring information.

11.Data security

11.1. Switch to the enhanced whitelist mode

This topic describes how to switch from the standard whitelist mode to the enhanced whitelist mode for an ApsaraDB RDS instance. The enhanced whitelist mode provides higher security.

Network isolation modes

ApsaraDB RDS instances support the following network isolation modes:

- Standard whitelist mode

  IP addresses from both the classic network and virtual private clouds (VPCs) are added to the same IP address whitelist. The standard whitelist mode may incur security risks. Therefore, we recommend that you switch the network isolation mode to enhanced whitelist.

- Enhanced whitelist mode

  An enhanced IP address whitelist can contain only the IP addresses from the classic network or VPCs. When you create an enhanced IP address whitelist, you must specify its network type.

Changes after you switch to the enhanced whitelist mode

- If your ApsaraDB RDS instance resides in a VPC, an IP address whitelist of the VPC network type is automatically created. The new IP address whitelist contains all IP addresses that are replicated from the original IP address whitelists.
- If your ApsaraDB RDS instance resides in the classic network, an IP address whitelist of the classic network type is automatically created. The new IP address whitelist contains all IP addresses that are replicated from the original IP address whitelists.
- If your ApsaraDB RDS instance runs in hybrid access mode, two identical IP address whitelists are created: an IP address whitelist of the VPC network type and an IP address whitelist of the classic network type. Both the new IP address whitelists contain all IP addresses that are replicated from the original IP address whitelists. For more information, see Hybrid access from both the classic network and VPCs.

> **Note** After you switch to the enhanced whitelist mode, the IP addresses that come from Elastic Compute Service (ECS) security groups remain unchanged.

Precautions

- You can switch from the standard whitelist mode to the enhanced whitelist mode for ApsaraDB RDS instances that use local SSDs, but not the other way around.
- In enhanced whitelist mode, an IP address whitelist of the classic network type can also be used to allow access over the Internet. If you want to access your ApsaraDB RDS instance from a host over the Internet, you must add the public IP address of the host to an IP address whitelist of the classic network type.

Procedure

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Data Security**.
5. On the **Whitelist Settings** tab, click **Switch to Enhanced Whitelist (Recommended)**.
6. In the message that appears, click **Confirm**.

11.2. Configure an IP address whitelist

This topic describes how to configure a whitelist for an ApsaraDB RDS instance. Only entities that are listed in a whitelist can access your ApsaraDB RDS instance.

Context

Whitelists make your ApsaraDB RDS instance more secure and do not interrupt the operations of your ApsaraDB RDS instance when you configure whitelists. We recommend that you perform maintenance on your whitelists on a regular basis.

To configure a whitelist, perform the following operations:

- Configure a whitelist: Add IP addresses to allow them to connect to the ApsaraDB RDS instance.

  > **Note** The IP address whitelist labeled default contains only the default IP address 0.0.0.0/0, which allows all entities to access your ApsaraDB RDS instance.

- Configure an ECS security group: Add an ECS security group for the ApsaraDB RDS instance to allow ECS instances in the group to connect to the ApsaraDB RDS instance.

Procedure

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Data Security**.
5. On the **Whitelist Settings** tab, click **Edit** corresponding to the **default** whitelist.

   > **Note** You can also click **Create Whitelist** to create a whitelist.

6. In the **Edit Whitelist** dialog box, enter the IP addresses or CIDR blocks used to access the instance and click **OK**. The following section describes the rules:
   - If you enter the CIDR block 10.10.10.0/24 in the IP Addresses field, all IP addresses in the 10.10.10.X format can access your ApsaraDB RDS instance.
   - If you enter more than one IP address or CIDR block, you must separate them with commas (,). Do not add spaces before or after the commas. Example: 192.168.0.1,172.16.213.9.
   - If you click **Add Internal IP Addresses of ECS Instances**, the IP addresses of all ECS instances created within your Alibaba Cloud account are displayed. You can select the required IP

addresses to add them to the IP address whitelist.

11.3. Configure SSL encryption

This topic describes how to configure SSL encryption for an ApsaraDB RDS instance.

Prerequisites

The ApsaraDB RDS instance uses standard SSDs.

Precautions

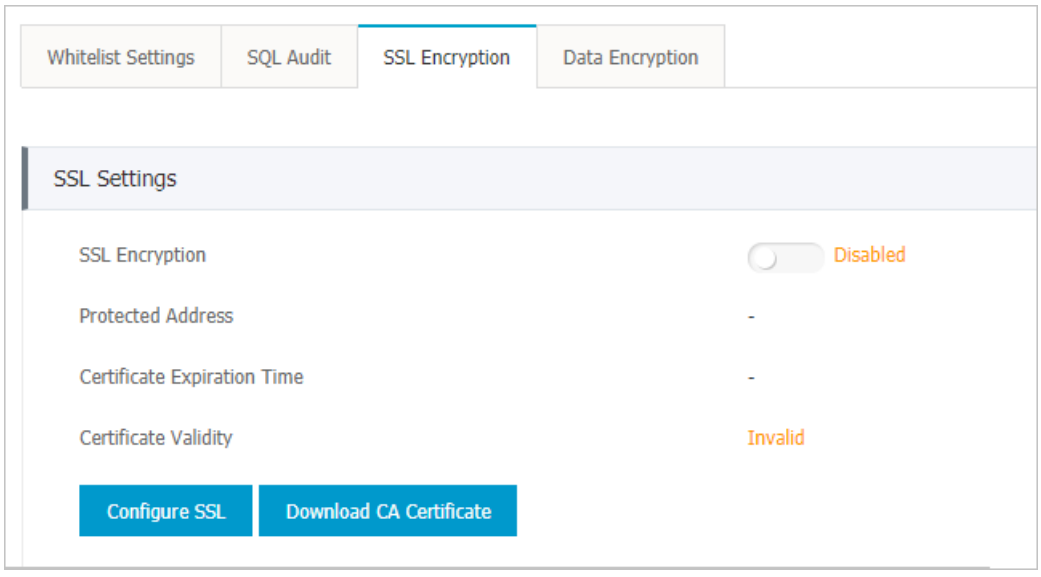
- After SSL encryption is enabled, SSL is used to encrypt all the data that is transmitted over an internal network or the Internet. SSL encryption protects the data in transit from being leaked.
- After SSL encryption is enabled, you must close the existing connection and establish a new one to bring SSL encryption into effect.

Enable SSL encryption

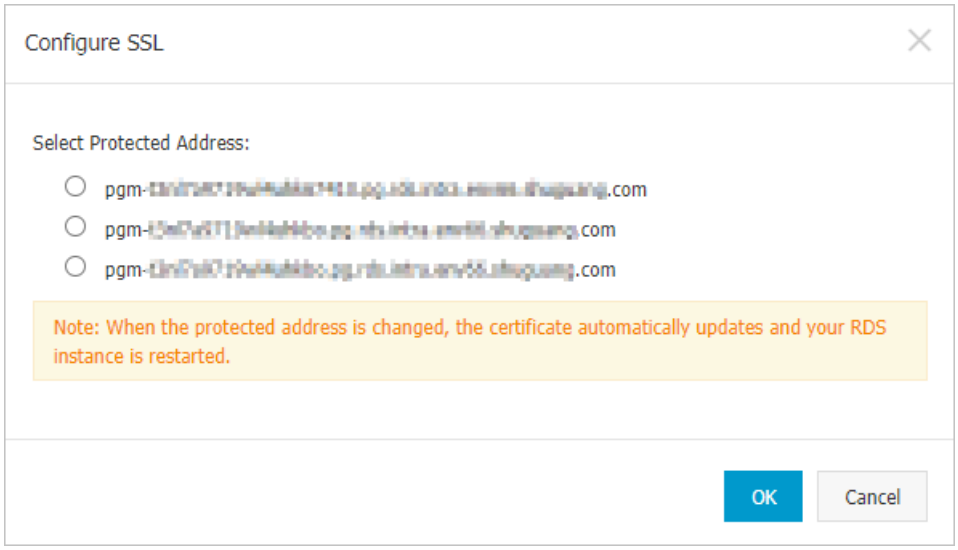
The Internet Engineering Task Force (IETF) upgraded SSL 3.0 to Transport Layer Security (TLS). However, the term "SSL encryption" is still used in the industry. In this topic, SSL encryption refers to TLS encryption.

> **Note** ApsaraDB RDS supports TLS 1.0, TLS 1.1, and TLS 1.2.

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the instance that you want to manage.
3. In the left-side navigation pane, click **Data Security**.
4. Click the **SSL Encryption** tab.



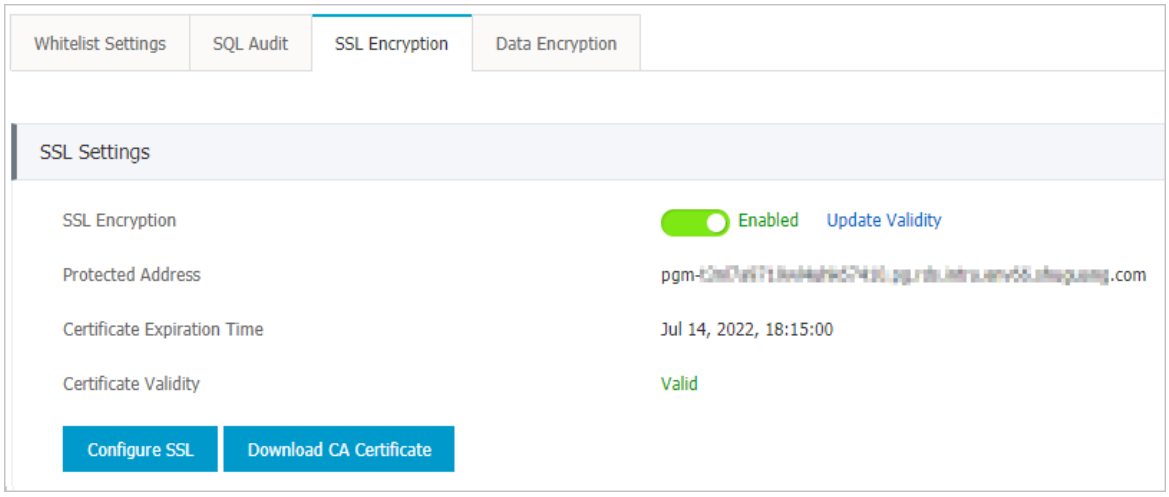
5. Click **Configure SSL**. In the dialog box that appears, select a protected endpoint, click OK, and then wait for the system to enable SSL encryption.



Note After SSL encryption is enabled, you must set the SSL mode to **Prefer** when you log on from your client.

Disable SSL encryption

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the instance that you want to manage.
3. In the left-side navigation pane, click **Data Security**.
4. Click the **SSL Encryption** tab.



5. Turn off SSL Encryption. In the message that appears, click OK and wait for the system to disable SSL encryption.

11.4. Configure data encryption

This topic describes how to configure data encryption for an ApsaraDB RDS instance that uses standard or enhanced SSDs. The disk encryption feature maximizes the protection for your data and eliminates the need to modify business or application configurations. ApsaraDB RDS automatically applies disk encryption to both the snapshots that are generated from the encrypted SSDs and the SSDs that are created from those snapshots.

Prerequisites

The storage type of the instance is standard SSD.

Configure disk encryption

1. Enter the URL of the Apsara Uni-manager Management Console in the address bar and press the Enter key.
2. Enter your username and password.

   Obtain the username and password that you use to log on to the Apsara Uni-manager Management Console from the operations administrator.

   > **Note** The first time that you log on to the Apsara Uni-manager Management Console, you must change the password of your account. For security purposes, your password must meet the minimum complexity requirements. The password must be 10 to 32 characters in length and must contain at least two of the following character types:
   >
   > - Uppercase or lowercase letters
   > - Digits
   > - Special characters, including exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%)

3. Click **Log On**.
4. In the top navigation bar, choose **Products** > Security > Key Management Service.
5. On the Keys page, click **Create Key**.
6. Configure the following parameters.

Section	Parameter	Description
Region	**Organization**	The organization to which the key belongs.
	Resource Set	The resource set to which the key belongs.
	Region	The region to which the key belongs.
Basic	**Key Type**	KMS supports the following key types: ○ Symmetric keys: ■ Aliyun_AES_256 ■ Aliyun_SM4 ○ Asymmetric keys: ■ RSA_2048 ■ EC_P256 ■ EC_P256K ■ EC_SM2

Basic Settings	Parameter	Description
	Key Purpose	ENCRYPT/DECRYPT: The purpose of the CMK is to encrypt or decrypt data.
	Protection Level	○ SOFTWARE: Use a software module to protect the CMK. ○ HSM: Host the CMK in a hardware security module (HSM). Managed HSM uses the HSM as dedicated hardware to safeguard the CMK.
	Alias	The identifier of the CMK. For more information, see Use aliases in *KMS User Guide*.
	Description	The description of the CMK.
Advanced Settings	**Rotation Period**	Specifies whether to enable automatic rotation. If you choose to enable automatic rotation, you must select a rotation period. For more information about rotation, see Key rotation in *KMS User Guide*. Valid values: ○ 30 Days ○ 90 Days ○ 180 Days ○ 365 Days ○ Custom: Customize a period that ranges from 7 to 730 days. **Note** You can specify this parameter only if Key Type is set to Aliyun_AES_256 or Aliyun_SM4.
	Key Material Source	The source of key material. ○ Key Management Service: Use KMS to generate key material. ○ External: Manually import external key material. **Note** If **Rotation Period** is set to **Enable**, the **External** option is unavailable.

7. Click **Submit**.
8. Create an ApsaraDB RDS instance with disk encryption enabled. For more information, see Create an ApsaraDB RDS for PostgreSQL instance that uses standard or enhanced SSDs.

12.Logs and audit

12.1. Configure SQL audit

This topic describes how to configure the SQL audit feature to audit SQL executions and check the details. SQL audit does not affect instance performance.

Precautions

- SQL audit does not affect instance performance.
- SQL audit logs are retained for 30 days.
- Log files exported from SQL audit are retained for two days. The system deletes files that are retained for longer than two days.
- SQL audit is disabled by default. You must manually enable it.
- You cannot view logs that are generated before SQL audit is enabled.

Enable SQL audit

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Data Security**.
5. Click the **SQL Audit** tab.
6. Click **Enable SQL Audit** or **Enable now**.
7. In the message that appears, click **Confirm**.

> **Note** After SQL audit is enabled, you can query SQL information based on conditions such as the time, database, user, and keyword.

Disable SQL audit

You can disable SQL audit when it is no longer needed. To disable SQL audit, perform the following steps:

> **Notice** After SQL audit is disabled, all SQL audit logs are deleted. We recommend that you export and store audit logs to your computer before you disable SQL audit.

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Data Security**.
5. Click the **SQL Audit** tab. Click **Export File**.

> **Note** If more than 1 million SQL audit logs meet the filter conditions you specify, only 1 million logs can be exported. SQL audit logs are exported at a speed of 900 entries per second. It takes about 20 minutes to export 1 million SQL audit logs.

6. Click **Files**. Find a file and click Download in the **Action** column to download the file to your computer.
7. Click **Disable SQL Audit**.
8. In the message that appears, click **Confirm**.

12.2. Manage logs

You can view logs for errors, slow queries, and primary/secondary instance switching for ApsaraDB RDS for PostgreSQL instances in the ApsaraDB RDS console or by executing SQL statements. These logs help you troubleshoot errors. This topic describes how to manage logs in the console.

Procedure

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Logs**.
5. On the **Logs** page, click the **Error Logs**, **Slow Query Logs**, or **Primary/Secondary Switching Logs** tab, select a time range, and then click **Search**.

Tab	Description
Error Logs	Records database running errors that occurred within the last month.
Slow Query Logs	Records SQL statements within the last month that took longer than one second to execute. Duplicated SQL statements are removed.
Primary/Secondary Switching Logs	Records switchovers between the primary and secondary instances within the last month.

13.Backup

13.1. Back up an ApsaraDB RDS for PostgreSQL instance

This topic describes how to back up an ApsaraDB RDS for PostgreSQL instance. You can configure a backup policy that is used to automatically back up your ApsaraDB RDS instance. If you do not configure a backup policy, the default backup policy is used. You can also manually back up your ApsaraDB RDS instance.

Precautions

- Do not execute data definition language (DDL) statements during a backup. These statements trigger locks on tables, and the backup may fail as a result of the locks.
- We recommend that you back up your ApsaraDB RDS instance during off-peak hours.
- If your ApsaraDB RDS instance has a large amount of data, a backup may require a long period of time.
- Backup files are retained for a specific retention period. Before the specific retention period elapses, we recommend that you download the required backup files to your computer.

Backup description

ApsaraDB RDS for PostgreSQL allows you to perform full physical backup and back up archived redo log files of databases.

Configure a backup policy for automatic backups

ApsaraDB RDS can automatically back up your instance based on the backup policy that you specify.

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Backup and Restoration**.
5. On the **Backup and Restoration** page, click the **Backup Settings** tab and click **Edit**.
6. In the dialog box that appears, configure the following parameters and click **OK**. The following table lists the parameters.

Parameter	Description
Data Retention Period	The number of days for which you want to retain data backup files. Valid values: 7 to 730. Unit: days. Default value: 7.

Parameter	Description
Backup Cycle	The cycle to create backups. You can select one or more days of the week. **Note** To ensure data security, we recommend that you back up your ApsaraDB RDS instance at least twice a week.
Backup Time	The period of time for which you want to back up data. Unit: hours.
Log Backup	Specifies whether to enable the log backup feature. **Notice** If you disable this feature, all log backup files are deleted and your instance cannot be restored to previous points in time.
Log Retention Period	○ The period of time for which you want to retain log backup files. Valid values: 7 to 730. Unit: days. Default value: 7. ○ The log retention period must be less than or equal to the data retention period.
OSS Dump Status	Specifies whether to enable Object Storage Service (OSS) dump. When OSS dump is enabled, new backup files are automatically dumped to a specific OSS bucket. Valid values: ○ Enabled ○ Disabled
OSS Dumped Data	The type of backup files that are dumped to an OSS bucket. You can select multiple values. Valid values: ○ Data Backup ○ Log Backup
OSS Bucket	The OSS bucket to which you want to dump backup files.

Manually back up your ApsaraDB RDS instance

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the upper-right corner of the page, click **Back Up Instance**. The **Back Up Instance** dialog box appears.
5. Select the backup mode and backup policy, and click **OK**.

> **Note** The backup mode is **Full Backup** and the backup policy is **Instance Backup**.

What's next

You can click the  icon in the upper-right corner of the page to view the task progress displayed in the **Task Progress** list.

13.2. Download data and log backup files

This topic describes how to download unencrypted data and log backup files in the ApsaraDB RDS console to archive the files and restore data to an on-premises database.

Procedure

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Backup and Restoration** to go to the **Backup and Restoration** page.
5. Click the Data Backup or Archived Logs tab.
   - To download data backup files, click the **Data Backup** tab.
   - To download log files, click the **Archived Logs** tab.
6. Select a time range to which you want to restore the instance.
7. Find the data backup or log file that you want to download and click **Download** in the **Actions** column.

   > **Note**
   > - If you want to use a data backup file to restore data, select the backup file that is the closest to the time for restoration.
   > - If you want to use a log file to restore data to an on-premises database, take note of the following items:
   >   - The instance No. of the log file must be the same as that of the data backup file.
   >   - The start time of the log file must be later than the data backup time and earlier than the time for restoration.

8. In the message that appears, select a download method.

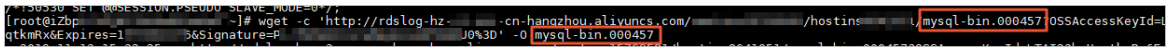
Download method	Description
Download	Download the file by using the public endpoint.

Download method	Description
Copy Internal Endpoint	Copy the internal endpoint to download the file. If your ECS and ApsaraDB RDS instances are deployed within the same region, you can log on to the ECS instance and use the internal endpoint to download the file. This method is fast and secure.
Copy Public Endpoint	Copy the public endpoint to download the file. If you want to use other tools to download the file, use the public endpoint.

Note If you use a Linux operating system, you can run the following command to download the file:

```
wget -c '<Public endpoint of the backup file, which is the download URL>' -O <File name>
```

- The -c option enables resumable download.
- The -O option saves the downloaded file by using a specified name. We recommend that you use the file name contained in the download URL.
- If the URL contains more than one parameter, enclose the download URL in a pair of single quotation marks (').



13.3. Create a logical backup for an ApsaraDB RDS for PostgreSQL instance

This topic describes how to use the pg_dump utility to create a logical backup for an ApsaraDB RDS for PostgreSQL instance and export the backup file to your computer.

Context

The pg_dump utility provided with PostgreSQL is used to back up individual databases. For more information, see pg_dump.

In this example, an RDS instance that runs PostgreSQL 10 and a host that runs CentOS 7 are used.

Prerequisites

- The IP address of your Elastic Compute Service (ECS) instance or on-premises host is added to the whitelist of an ApsaraDB RDS for PostgreSQL instance. For more information, see Configure an IP address whitelist.
- Your ECS instance or on-premises host runs the same version of PostgreSQL as your ApsaraDB RDS for PostgreSQL instance.

Precautions

We recommend that you use the privileged account of the RDS instance. This ensures that you have all the required permissions.

Back up a database

1. Log on to your ECS instance or on-premises host. Then, run the following command to back up a database of the RDS instance:

```
pg_dump -h '<hostname>' -U <username> -p <port> -Fc <dbname> > <dumpdir>
```

Parameter	Description
hostname	The endpoint that is used to connect to the ApsaraDB RDS for PostgreSQL instance. **Note** If your ECS instance uses an internal endpoint to connect to the RDS instance, you must make sure that the ECS instance and the RDS instance use the same network type. If both instances use the VPC network type, you must also make sure that they reside in the same VPC. For more information about how to view the internal endpoint, see View and modify the internal endpoint and port number.
username	The username of the privileged account of the ApsaraDB RDS for PostgreSQL instance.
port	The port number that is used to connect to the ApsaraDB RDS for PostgreSQL instance.
-Fc	The format of the output file. `-Fc` specifies the use of the custom format. This format is ideal when you use pg_restore to import logical backup files and restore databases. For more information, see pg_dump.
dbname	The name of the database that you want to back up.
dumpdir	The directory and name of the logical backup file to export.

Example

```
pg_dump -h 'pgm-bpxxxxxx.pg.rds.aliyuncs.com' -U test123 -p 3433 -Fc testdb > /tmp/testdb.dump
```

2. When `Password:` appears on the command line, enter the password of the privileged account of the RDS instance and press the Enter key.

Back up one or more tables

1. Log on to your ECS instance or on-premises host. Then, run the following command to back up one or more tables from a database in the RDS instance:

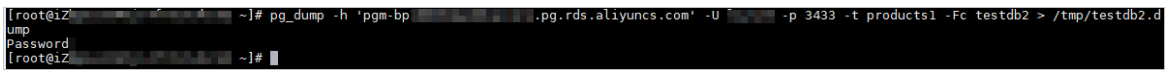
```
pg_dump -h '<hostname>' -U <username> -p <port> -t <table> -Fc <dbname> > <dumpdir>
```

Parameter	Description
hostname	The endpoint that is used to connect to the ApsaraDB RDS for PostgreSQL instance. **Note** If your ECS instance uses an internal endpoint to connect to the RDS instance, you must make sure that the ECS instance and the RDS instance use the same network type. If both instances use the VPC network type, you must also make sure that they reside in the same VPC. For more information about how to view the internal endpoint, see View and modify the internal endpoint and port number.
username	The username of the privileged account of the ApsaraDB RDS for PostgreSQL instance.
port	The port number that is used to connect to the ApsaraDB RDS for PostgreSQL instance.
table	The name of the table that you want to back up. You can use `-t <table>` to specify multiple tables.
-Fc	The format of the output file. `-Fc` specifies the use of the custom format. This format is ideal when you use pg_restore to import logical backup files and restore databases. For more information, see pg_dump.
dbname	The name of the database that you want to back up.
dumpdir	The directory and name of the logical backup file to export.

Example

```
pg_dump -h 'pgm-bpxxxxxx.pg.rds.aliyuncs.com' -U test123 -p 3433 -t products1 -Fc testd
b2 > /tmp/testdb2.dump
```

2. When `Password:` appears on the command line, enter the password of the privileged account of the RDS instance and press the Enter key.



Back up a database with one or more tables excluded

1. Log on to your ECS instance or on-premises host. Then, run the following command to back up a database from the RDS instance with one or more tables excluded:

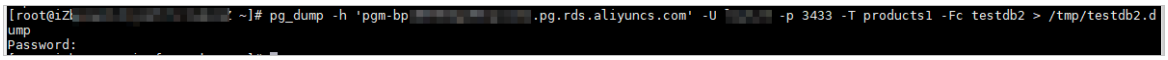
```
pg_dump -h '<hostname>' -U <username> -p <port> -T <table> -Fc <dbname> > <dumpdir>
```

Parameter	Description
hostname	The endpoint that is used to connect to the ApsaraDB RDS for PostgreSQL instance. **Note** If your ECS instance uses an internal endpoint to connect to the RDS instance, you must make sure that the ECS instance and the RDS instance use the same network type. If both instances use the VPC network type, you must also make sure that they reside in the same VPC. For more information about how to view the internal endpoint, see View and modify the internal endpoint and port number.
username	The username of the privileged account of the ApsaraDB RDS for PostgreSQL instance.
port	The port number that is used to connect to the ApsaraDB RDS for PostgreSQL instance.
table	The name of the table that you want to exclude. You can use `-T <table>` to specify multiple tables.
-Fc	The format of the output file. `-Fc` specifies the use of the custom format. This format is ideal when you use pg_restore to import logical backup files and restore databases. For more information, see pg_dump.
dbname	The name of the database that you want to back up.
dumpdir	The directory and name of the logical backup file to export.

Example

```
pg_dump -h 'pgm-bpxxxxx.pg.rds.aliyuncs.com' -U test123 -p 3433 -T products1 -Fc testdb
2 > /tmp/testdb2.dump
```

2. When `Password:` appears on the command line, enter the password of the privileged account of the RDS instance and press the Enter key.



Back up the schema of a database with data excluded

1. Log on to your ECS instance or on-premises host. Then, run the following command to back up the schema of a database from the RDS instance:

```
pg_dump -h '<hostname>' -U <username> -p <port> -s -Fc <dbname> > <dumpdir>
```

Parameter	Description

Parameter	Description
hostname	The endpoint that is used to connect to the ApsaraDB RDS for PostgreSQL instance. **Note** If your ECS instance uses an internal endpoint to connect to the RDS instance, you must make sure that the ECS instance and the RDS instance use the same network type. If both instances use the VPC network type, you must also make sure that they reside in the same VPC. For more information about how to view the internal endpoint, see View and modify the internal endpoint and port number.
username	The username of the privileged account of the ApsaraDB RDS for PostgreSQL instance.
port	The port number that is used to connect to the ApsaraDB RDS for PostgreSQL instance.
-s	Specifies that only the schema of the database is backed up. The data of the database is not backed up. For more information, see pg_dump.
-Fc	The format of the output file. `-Fc` specifies the use of the custom format. This format is ideal when you use pg_restore to import logical backup files and restore databases. For more information, see pg_dump.
dbname	The name of the database that you want to back up.
dumpdir	The directory and name of the logical backup file to export.

Example

```
pg_dump -h 'pgm-bpxxxxx.pg.rds.aliyuncs.com' -U test123 -p 3433 -s -Fc testdb2 > /tmp/testdb2.dump
```

2. When `Password:` appears on the command line, enter the password of the privileged account of the RDS instance and press the Enter key.

13.4. Create a full backup of an ApsaraDB RDS for PostgreSQL instance

This topic describes how to use the pg_basebackup utility provided by open source PostgreSQL to create a full backup of your ApsaraDB RDS for PostgreSQL instance and export the backup files to your computer.

Prerequisites

- The IP address of your ECS instance or host is added to a whitelist of your ApsaraDB RDS for PostgreSQL instance. For more information, see Configure an IP address whitelist.
- Your ECS instance or host runs the same version of PostgreSQL as the ApsaraDB RDS for PostgreSQL instance.

Context

pg_basebackup backs up all data of a PostgreSQL instance. Backup files can be used for point-in-time recovery. For more information, visit pg_basebackup.

In this example, CentOS 7 is used to create a full backup.

Precautions

We recommend that you use the privileged account of the ApsaraDB RDS for PostgreSQL instance to ensure that you have all the required permissions.

Procedure

Note pg_basebackup cannot back up a single database or database object. For more information about how to back up a single database or database object, see Create a logical backup for an ApsaraDB RDS for PostgreSQL instance.

1. Log on to your ECS instance or host. Then, run the following command to back up a database from the ApsaraDB RDS for PostgreSQL instance:

   ```
   pg_basebackup -Ft -Pv -Xf -z -D <backupdir> -Z5 -h '<hostname>' -p <port> -U <username>
   -W
   ```

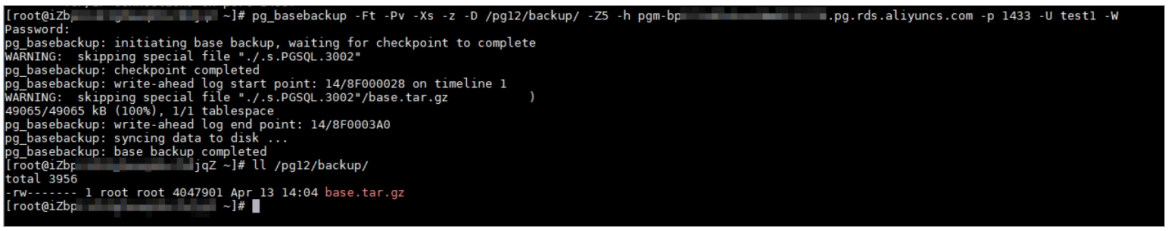
   The following table describes the parameters in this command. For more information, visit pg_basebackup.

Parameter	Description
backupdir	The directory of backup files that are exported. The system automatically creates this directory. However, if this directory already exists and is not empty, the system reports an error.
hostname	The internal endpoint of the ApsaraDB RDS for PostgreSQL instance. For more information about how to view the internal endpoint, see View and modify the internal endpoint and port number.
port	The port number of the ApsaraDB RDS for PostgreSQL instance.
username	A username of the ApsaraDB RDS for PostgreSQL instance.

   Example:

```
pg_basebackup -Ft -Pv -Xf -z -D /pg12/backup1/ -Z5 -h pgm-bpxxxxx.pg.rds.aliyuncs.com -
p 1433 -U test1 -W
```

2. When `Password:` appears, enter the password of the username of the ApsaraDB RDS for PostgreSQL instance and press the Enter key.



14.Restoration

14.1. Restore data of an ApsaraDB RDS for PostgreSQL instance

This topic describes how to use the backup data of an ApsaraDB RDS for PostgreSQL instance to restore data.

Precautions

- The new instance must have the same whitelist, backup, and parameter settings as the original instance.
- The new instance must have the same data and account information as the backup set or instance at the time point.

Prerequisites

The original instance must meet the following requirements:

- The original instance is in the Running state and is not locked.
- The original instance does not have ongoing migration tasks.
- If you want to restore data to a point in time, the log backup feature is enabled for the original instance.
- If you want to restore an instance from a backup set, the original instance has at least one backup set.

Restore data of an ApsaraDB RDS for PostgreSQL instance

1. Log on to the ApsaraDB RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the Actions column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Backup and Restoration**.
5. In the upper-right corner of the page, click **Restore Database (Previously Clone Database)**.
6. Configure the following parameters.

Section	Parameter	Description
Region	**Region**	The region where the instance is deployed.

Section	Parameter	Description
Restore Database	Restore Mode	◦ **By Time**: You can restore data to a point in time within the retention period of the log backup. For more information about how to view or change the retention period of log backups, see Back up an ApsaraDB RDS for PostgreSQL instance. ◦ **By Backup Set** **Note** The **By Time** option appears only when the log backup feature is enabled.
	Restore Time	The time to which the database is restored. This parameter is displayed when you set **Restore Mode** to **By Time**.
	Backup Set	The backup set used to restore the database. This parameter is displayed when you set **Restore Mode** to **By Backup Set**.
Specifications	Instance Name	The name of the instance.
	Database Engine	The engine of the database. The value of this parameter is set to **PostgreSQL** and cannot be changed.
	Engine Version	The version of the database engine. The value of this parameter is set to the engine version of the current instance and cannot be changed.
	Edition	The edition of the instance.
	Storage Type	The storage type of the instance. The value of this parameter is set to the storage type of the current instance and cannot be changed.
	Instance Type	The instance type of the instance. Memory size determines the maximum number of connections and IOPS. The actual values are displayed in the console. For more information, see Instance types in *Instance types* of *ApsaraDB RDS Product Introduction*.
	Storage Capacity	The storage capacity of the instance, including the space to store data, system files, binlog files, and transaction files. Valid values: 20 to 600. Unit: GB. The value must be in 1 GB increments.

Section	Parameter	Description
Network Type	**Network Type**	The network type of the instance. ApsaraDB RDS instances support the following network types: ○ **Classic Network**: Cloud services on a classic network are not isolated from each other. Unauthorized access to a cloud service is blocked only by the security group or whitelist policy of the service. ○ **VPC**: A virtual private cloud (VPC) helps you build an isolated network environment on Apsara Stack. You can customize route tables, CIDR blocks, and gateways in a VPC. We recommend that you select VPC for improved security. **Note** If you set the network type to **VPC**, you must also select a **VPC** and a **vSwitch**.

7. Click **Submit**.

14.2. Restore data from a logical backup file

This topic describes how to restore data from a logical backup file to an ApsaraDB RDS for PostgreSQL instance or an on-premises PostgreSQL database.

Context

A logical backup file is used to restore a small amount of data, such as data in a table. For a large amount of data, we recommend that you restore it from a full physical backup file to a new ApsaraDB RDS instance and then use Data Transmission Service (DTS) to migrate data to the original ApsaraDB RDS instance.

Prerequisites

Data in the ApsaraDB RDS for PostgreSQL instance is logically backed up. For more information, see Create a logical backup for an ApsaraDB RDS for PostgreSQL instance.

Precautions

- We recommend that you do not restore data to the default postgres database.
- When you restore the data of a table, the system does not restore the database objects on which the table depends. The restoration may fail.

Restore the data of a database

1. Log on to the ECS instance or on-premises host that houses the logical backup file and run the following command to restore the data of a database:

```
pg_restore -h '<hostname>' -U <username> -p <port> -d <dbname> <dumpdir>
```

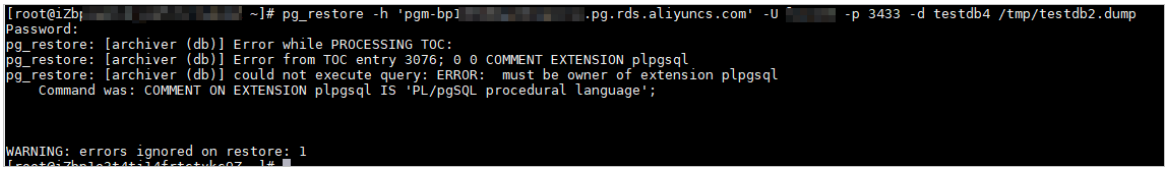
Parameter	Description
hostname	The endpoint of the ApsaraDB RDS for PostgreSQL instance. **Note** If your ECS instance connects to the ApsaraDB RDS for PostgreSQL instance by using an internal endpoint, make sure that the ECS and ApsaraDB RDS instances have the same network type. If both instances are of the VPC network type, make sure that both instances reside within the same VPC. For more information about how to view the internal endpoint, see View and modify the internal endpoint and port number.
username	The username of the privileged account of the ApsaraDB RDS for PostgreSQL instance.
port	The port number of the ApsaraDB RDS for PostgreSQL instance.
dbname	The name of the database whose data you want to restore.
dumpdir	The directory and name of the logical backup file to use.

Example:

```
pg_restore -h 'pgm-bpxxxxx.pg.rds.aliyuncs.com' -U test123 -p 3433 -d testdb2 /tmp/test
db.dump
```

2. When `Password:` appears, enter the password of the privileged account of your ApsaraDB RDS instance and press the Enter key.

Note You can ignore alerts generated by the embedded plpgsql plug-in.



Restore the data of a table

1. Log on to the ECS instance or on-premises host that houses the logical backup file and run the following command to restore the data of a table:

```
pg_restore -h '<hostname>' -U <username> -p <port> -d <dbname> -t <table> -c <dumpdir>
```

Parameter	Description

Parameter	Description
hostname	The endpoint of the ApsaraDB RDS for PostgreSQL instance. **Note** If your ECS instance connects to the ApsaraDB RDS for PostgreSQL instance by using an internal endpoint, make sure that the ECS and ApsaraDB RDS instances have the same network type. If both instances are of the VPC network type, make sure that both instances reside within the same VPC. For more information about how to view the internal endpoint, see View and modify the internal endpoint and port number.
username	The username of the privileged account of the ApsaraDB RDS for PostgreSQL instance.
port	The port number of the ApsaraDB RDS for PostgreSQL instance.
dbname	The name of the database whose data you want to restore.
table	The name of the table whose data you want to restore.
-c	`-c`: specifies to delete the database objects on which the table depends before data restoration. For more information, visit pg_restore.
dumpdir	The directory and name of the logical backup file to use.

Example:

```
pg_restore -h 'pgm-bpxxxxx.pg.rds.aliyuncs.com' -U test123 -p 3433 -d testdb2 -t produc
ts -c /tmp/testdb.dump
```

2. When `Password:` appears, enter the password of the privileged account of your ApsaraDB RDS instance and press the Enter key.

Restore the schema of a database with data excluded

1. Log on to the ECS instance or on-premises host that houses the logical backup file and run the following command to restore only the schema of a database:

```
pg_restore -h '<hostname>' -U <username> -p <port> -d <dbname> -s <dumpdir>
```

Parameter	Description

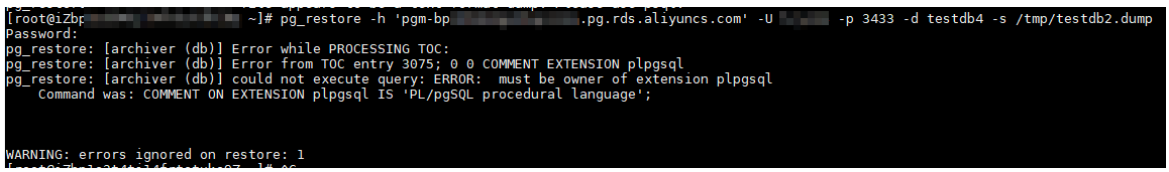
Parameter	Description
hostname	The endpoint of the ApsaraDB RDS for PostgreSQL instance. **Note** If your ECS instance connects to the ApsaraDB RDS for PostgreSQL instance by using an internal endpoint, make sure that the ECS and ApsaraDB RDS instances have the same network type. If both instances are of the VPC network type, make sure that both instances reside within the same VPC. For more information about how to view the internal endpoint, see View and modify the internal endpoint and port number.
username	The username of the privileged account of the ApsaraDB RDS for PostgreSQL instance.
port	The port number of the ApsaraDB RDS for PostgreSQL instance.
dbname	The name of the database whose schema you want to restore.
-s	`-s` : specifies to restore only the schema of the database. The data of the database is not restored. For more information, visit pg_restore.
dumpdir	The directory and name of the logical backup file to use.

Example:

```
pg_restore -h 'pgm-bpxxxxx.pg.rds.aliyuncs.com' -U test123 -p 3433 -d testdb4 -s /tmp/testdb2.dump
```

2. When `Password:` appears, enter the password of the privileged account of your ApsaraDB RDS instance and press the Enter key.

Note You can ignore alerts generated by the embedded plpgsql plug-in.



15.CloudDBA

15.1. Introduction to CloudDBA

CloudDBA is a cloud service for database self-detection, self-repair, self-optimization, self-maintenance, and self-security based on machine learning and expertise. CloudDBA helps you ensure stable, secure, and efficient databases without worrying about the management complexity and service failures caused by manual operations.

Features

In ApsaraDB RDS for PostgreSQL, CloudDBA provides the following features:

- Diagnostics

  You can diagnose your instance and view the visualized diagnostic results.

- Session management

  You can view sessions, check session statistics, analyze SQL statements, and optimize the execution of SQL statements.

- Real-time monitoring

  You can view the real-time information of your instance, such as the queries per second (QPS), transactions per second (TPS), number of connections, and network traffic.

- Storage analysis

  You can view the storage overview, trends, exceptions, tablespaces, and data spaces.

- Dashboard

  You can view and compare performance trends, customize monitoring dashboards, check exceptions, and view instance topologies.

- Slow query logs

  You can view the trends and statistics of slow queries.

15.2. Diagnostics

In ApsaraDB RDS for PostgreSQL, CloudDBA provides the diagnostics feature. This feature diagnoses your ApsaraDB RDS for PostgreSQL instance and visualizes the results.

Navigate to the Diagnostics tab

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the instance that you want to manage.
3. In the left-side navigation pane, choose **CloudDBA > Diagnostics**.
4. Click the **Diagnostics** tab.

> **Note** For more information, see Diagnostics in *Database Autonomy Service User Guide*.

15.3. Session management

In ApsaraDB RDS for PostgreSQL, CloudDBA provides the session management feature. This feature allows you to view and manage the sessions of an instance.

Navigate to the Session Management tab

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the instance that you want to manage.
3. In the left-side navigation pane, choose **CloudDBA > Diagnostics**.
4. Click the **Session Management** tab.

> **Note** For more information, see Instance sessions in *Database Autonomy Service User Guide*.

15.4. Real-time monitoring

In ApsaraDB RDS for PostgreSQL, CloudDBA provides the real-time monitoring feature. This feature allows you to view the real-time performance of your ApsaraDB RDS for PostgreSQL instance.

Navigate to the Real-time Monitoring tab

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the instance that you want to manage.
3. In the left-side navigation pane, choose **CloudDBA > Diagnostics**.
4. Click the **Real-time Monitoring** tab.

> **Note** For more information, see Real-time monitoring in *Database Autonomy Service User Guide*.

15.5. Storage analysis

In ApsaraDB RDS for PostgreSQL, CloudDBA provides the storage analysis feature. This feature allows you to check and solve storage exceptions in a timely manner to ensure database stability.

Context

You can use the storage analysis feature of CloudDBA to view the disk space usage of your ApsaraDB RDS for PostgreSQL instance and the number of remaining days when disk space is available. It also provides information about the space usage, fragmentation, and exception diagnostic results of a table.

Navigate to the Storage Analysis tab

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the instance that you want to manage.
3. In the left-side navigation pane, choose **CloudDBA > Diagnostics**.

4. Click the **Storage Analysis** tab.

> **Note** For more information, see Storage analysis in *Database Autonomy Service User Guide*.

15.6. Dashboard

In ApsaraDB RDS for PostgreSQL, CloudDBA provides the dashboard feature. This feature allows you to view performance trends in specific ranges, compare performance trends, and customize charts to view performance trends.

Navigate to the Dashboard page

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the instance that you want to manage.
3. In the left-side navigation pane, choose **CloudDBA > Dashboard**.

> **Note** For more information, see Dashboard in *Database Autonomy Service User Guide*.

15.7. Slow query logs

In ApsaraDB RDS for PostgreSQL, CloudDBA provides the slow query logs feature. This feature allows you to view the trends and execution details of slow queries and obtain optimization suggestions for your ApsaraDB RDS for PostgreSQL instance.

Navigate to the Slow Query Logs page

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the instance that you want to manage.
3. In the left-side navigation pane, choose **CloudDBA > Slow Query Logs**.

> **Note** For more information, see Slow query logs in *Database Autonomy Service User Guide*.

16.Plug-ins

16.1. Plug-ins supported

This topic describes the plug-ins that are supported by ApsaraDB RDS for PostgreSQL and their available versions.

PostgreSQL 12

Plug-in	Version
btree_gin	1.3
btree_gist	1.5
citext	1.6
cube	1.4
dblink	1.2
dict_int	1
earthdistance	1.1
fuzzystrmatch	1.1
hstore	1.6
intagg	1.1
intarray	1.2
isn	1.2
ltree	1.1
pg_buffercache	1.3

pg_prewarm	1.2
pg_stat_statements	1.7
pg_trgm	1.4
pgcrypto	1.3
pgrowlocks	1.2
pgstattuple	1.5
postgres_fdw	1
sslinfo	1.2
tablefunc	1
unaccent	1.1
plpgsql	1
plperl	1
pg_roaringbitmap	0.5.0
rdkit	3.8
mysql_fdw	1.1
ganos_geometry_sfcgal	3.0
ganos_geometry_topology	3.0

ganos_geometry	3.0
ganos_networking	3.0
ganos_pointcloud_geometry	3.0
ganos_pointcloud	3.0
ganos_raster	3.0
ganos_spatialref	3.0
ganos_trajectory	3.0
ganos_tiger_geocoder	3.0
ganos_address_standardizer	3.0
ganos_address_standardizer_data_us	3.0
wal2json	2.0
hll	2.14
plproxy	2.9.0
tsm_system_rows	1.0
tsm_system_time	1.0
smlar	1.0
tds_fdw	1.0
bigm	1.2

timescaledb	1.7.1

PostgreSQL 11

Plug-in	Version
plpgsql	1
pg_stat_statements	1.6
btree_gin	1.3
btree_gist	1.5
citext	1.5
cube	1.4
rum	1.3
dblink	1.2
dict_int	1
earthdistance	1.1
hstore	1.5
intagg	1.1
intarray	1.2
isn	1.2
ltree	1.1

pgcrypto	1.3
pgrowlocks	1.2
pg_prewarm	1.2
pg_trgm	1.4
postgres_fdw	1
sslinfo	1.2
tablefunc	1
timescaledb	1.7.1
unaccent	1.1
fuzzystrmatch	1.1
pgstattuple	1.5
pg_buffercache	1.3
zhparser	1
pg_pathman	1.5
plperl	1
orafce	3.8
pg_concurrency_control	1
varbitx	1

postgis	2.5.1
pgrouting	2.6.2
postgis_sfcgal	2.5.1
postgis_topology	2.5.1
address_standardizer	2.5.1
address_standardizer_data_us	2.5.1
ogr_fdw	1
ganos_pointcloud	3.0
ganos_spatialref	3.0
log_fdw	1.0
wal2json	2.2
PL/v8	2.3.13
pg_cron	1.1
pase	0.0.1
hll	2.14
oss_fdw	1.1
tds_fdw	2.0.1
plproxy	2.9.0

tsm_system_rows	1.0
tsm_system_time	1.0
smlar	1.0
zombodb	4.0
bigm	1.2

PostgreSQL 10

Plug-in	Version
pg_stat_statements	1.6
btree_gin	1.2
btree_gist	1.5
chkpass	1
citext	1.4
cube	1.2
dblink	1.2
dict_int	1
earthdistance	1.1
hstore	1.4
intagg	1.1

intarray	1.2
isn	1.1
ltree	1.1
pgcrypto	1.3
pgrowlocks	1.2
pg_prewarm	1.1
pg_trgm	1.3
postgres_fdw	1
sslinfo	1.2
tablefunc	1
unaccent	1.1
postgis_sfcgal	2.5.1
postgis_topology	2.5.1
fuzzystrmatch	1.1
postgis_tiger_geocoder	2.5.1
address_standardizer	2.5.1
address_standardizer_data_us	2.5.1
ogr_fdw	1

plperl	1
plv8	1.4.2
plls	1.4.2
plcoffee	1.4.2
uuid-ossp	1.1
zhparser	1
pgrouting	2.6.2
pg_hint_plan	1.3.0
pgstattuple	1.5
oss_fdw	1.1
ali_decoding	0.0.1
varbitx	1
pg_buffercache	1.3
q3c	1.5.0
pg_sphere	1
smlar	1
rum	1.3
pg_pathman	1.5

aggs_for_arrays	1.3.1
mysql_fdw	1
orafce	3.6
plproxy	2.8.0
pg_concurrency_control	1
postgis	2.5.1
ganos_geometry_sfcgal	2.2
ganos_geometry_topology	2.2
ganos_geometry	2.2
ganos_networking	2.2
ganos_pointcloud_geometry	2.2
ganos_pointcloud	2.2
ganos_raster	2.2
ganos_spatialref	2.2
ganos_trajectory	2.2
ganos_tiger_geocoder	2.2
ganos_address_standardizer	2.2
ganos_address_standardizer_data_us	2.2

PostgreSQL 9.4

Plug-in	Version
plpgsql	1
pg_stat_statements	1.2
btree_gin	1
btree_gist	1
chkpass	1
citext	1
cube	1
dblink	1.1
dict_int	1
earthdistance	1
hstore	1.3
intagg	1
intarray	1
isn	1
ltree	1
pgcrypto	1.1

pgrowlocks	1.1
pg_prewarm	1
pg_trgm	1.1
postgres_fdw	1
sslinfo	1
tablefunc	1
tsearch2	1
unaccent	1
postgis	2.2.8
postgis_topology	2.2.8
fuzzystrmatch	1
postgis_tiger_geocoder	2.2.8
plperl	1
pltcl	1
plv8	1.4.2
plls	1.4.2
plcoffee	1.4.2
uuid-ossp	1

zhparser	1
pgrouting	2.0.0
rdkit	3.4
pg_hint_plan	1.1.3
pgstattuple	1.2
oss_fdw	1.1
jsonbx	1
ali_decoding	0.0.1
varbitx	1
pg_buffercache	1
smlar	1
pg_sphere	1
q3c	1.5.0
pg_awr	1
imgsmlr	1
orafce	3.6
pg_concurrency_control	1

16.2. Use mysql_fdw to read data from and write data to a MySQL database

This topic describes how to use the mysql_fdw plug-in of ApsaraDB RDS for PostgreSQL to read data from and write data to a database on an ApsaraDB RDS for MySQL instance or a self-managed MySQL database.

Prerequisites

- The instance runs PostgreSQL 10.
- Communication between your ApsaraDB RDS for PostgreSQL instance and the MySQL database is normal.

Context

PostgreSQL 9.6 and later support parallel computing. PostgreSQL 11 can use joins on up to a billion data records to complete queries in seconds. A number of users prefer to use PostgreSQL to build small-sized data warehouses and process highly concurrent access requests. PostgreSQL 13 is under development. It will support columnar storage engines that further improve analysis capabilities.

The mysql_fdw plug-in establishes a connection to synchronize data from a MySQL database to your ApsaraDB RDS for PostgreSQL instance.

Procedure

1. Log on to a database of your ApsaraDB RDS for PostgreSQL instance. For more information, see Connect to an ApsaraDB RDS for PostgreSQL instance.
2. Create the mysql_fdw plug-in.

   ```
   create extension mysql_fdw;
   ```

3. Define a MySQL server.

   ```
   CREATE SERVER <Name of the MySQL server>
   FOREIGN DATA WRAPPER mysql_fdw
   OPTIONS (host '<Endpoint used to connect to the MySQL server>', port '<Port used to con
   nect to the MySQL server>');
   ```

   Example:

   ```
   CREATE SERVER mysql_server
   FOREIGN DATA WRAPPER mysql_fdw
   OPTIONS (host 'rm-xxx.mysql.rds.aliyuncs.com', port '3306');
   ```

4. Map the MySQL server to an account created on your ApsaraDB RDS for PostgreSQL instance. Then, the account can be used to access data in the MySQL database on the MySQL server.

```
CREATE USER MAPPING FOR <Username of the account to which the MySQL server is mapped>

SERVER <Name of the MySQL server>
OPTIONS (username '<Username used to log on to the MySQL database>', password '<Passwor
d used to log on to the MySQL database>');
```

Example:

```
CREATE USER MAPPING FOR pgtest
SERVER mysql_server
OPTIONS (username 'mysqltest', password 'Test1234!') ;
```

5. Create a foreign MySQL table by using the account that you mapped to the MySQL server in the previous step.

> **Note** The field names in the foreign MySQL table must be the same as those in the table of the MySQL database. You can choose to create only the fields you want to query. For example, if the table in the MySQL database contains the ID, NAME, and AGE fields, you can create only the ID and NAME fields in the foreign MySQL table.

```
CREATE FOREIGN TABLE <Name of the foreign MySQL table> (<Name of Field 1> <Data type of
Field 1>,<Name of Field 2> <Data type of Field 2>...) server <Name of the MySQL server>
options (dbname '<Name of the MySQL database>', table_name '<Name of the table in the M
ySQL database>');
```

Example:

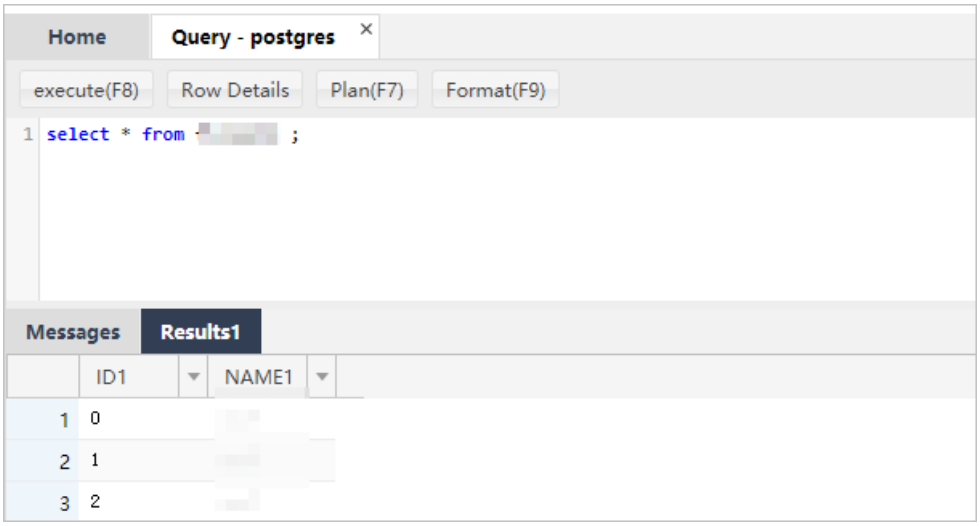
```
CREATE FOREIGN TABLE ft_test (id1 int, name1 text) server mysql_server options (dbname
'test123', table_name 'test');
```

What to do next

You can use the foreign MySQL table to test the performance of read and write operations on the MySQL database.

> **Note** Data can be written to the table in the MySQL database only when the table is assigned a primary key. If the table is not assigned a primary key, the following error is reported:
>
> ```
> ERROR:  first column of remote table must be unique for INSERT/UPDATE/DELETE operation.
> ```

```
select * from ft_test ;
insert into ft_test values (2,'abc');
insert into ft_test select generate_series(3,100),'abc';
select count(*) from ft_test ;
```



Run `postgres=> explain verbose select count(*) from ft_test;` to find out how the requests sent from your ApsaraDB RDS for PostgreSQL instance are executed to query data from the MySQL database. Command output:

```
                                QUERY PLAN
---------------------------------------------------------------------------
 Aggregate  (cost=1027.50..1027.51 rows=1 width=8)
   Output: count(*)
   ->  Foreign Scan on public.ft_test  (cost=25.00..1025.00 rows=1000 width=0)
         Output: id, info
         Remote server startup cost: 25
         Remote query: SELECT NULL FROM `test123`.`test`
(6 rows)
```

16.3. Use oss_fdw to read and write foreign data files

This topic describes how to use the oss_fdw plug-in to load data between Object Storage Service (OSS) and PostgreSQL or PPAS databases.

oss_fdw parameters

The oss_fdw plug-in uses a method similar to other Foreign Data Wrapper (FDW) interfaces to encapsulate foreign data stored in OSS. You can use oss_fdw to read data stored in OSS. This process is similar to reading data tables. oss_fdw provides unique parameters to connect and parse file data in OSS.

> **Note**
>
> - oss_fdw can read and write files of the following types in OSS: TXT and CSV files as well as GZIP-compressed TXT and CSV files.
> - The value of each parameter must be enclosed in double quotation marks (") and cannot contain unnecessary spaces.

CREATE SERVER parameters

- ossendpoint: the endpoint used to access OSS over the internal network, also known as the host.
- id oss: the AccessKey ID of the OSS account.
- key oss: the AccessKey secret of the OSS account.
- bucket: the bucket where the data you want to access is stored. You must create an OSS account before you specify this parameter.

The following fault tolerance parameters can be used for data import and export. If network connectivity is poor, you can adjust these parameters to ensure successful import and export.

- oss_connect_timeout: the connection timeout period. Default value: 10. Unit: seconds.
- oss_dns_cache_timeout: the DNS timeout period. Default value: 60. Unit: seconds.
- oss_speed_limit: the minimum data transmission rate. Default value: 1024. Unit: bytes/s.
- oss_speed_time: the maximum waiting period during which the data transmission rate is lower than the minimum value. Default value: 15. Unit: seconds.

If the default values of oss_speed_limit and oss_speed_time are used, a timeout error occurs when the transmission rate is lower than 1,024 bytes/s for 15 consecutive seconds.

CREATE FOREIGN TABLE parameters

- filepath: a file name that contains a path in OSS.
  - The file name specified by this parameter contains the directory name but not the bucket name.
  - This parameter matches multiple files in the corresponding path in OSS. You can load multiple files to a database.
  - You can import files that adhere to the filepath or filepath.x format to a database. The values of x must be consecutive numbers starting from 1.

    For example, among the files named filepath, filepath.1, filepath.2, filepath.3, and filepath.5, the first four files are matched and imported. The filepath.5 file is not imported.
- dir: the virtual file directory in OSS.
  - The specified directory must end with a forward slash (/).
  - All files (excluding subfolders and files in subfolders) in the virtual file directory specified by dir are matched and imported to a database.
- prefix: the prefix of the path name corresponding to the data file. The prefix does not support regular expressions. The prefix, filepath, and dir parameters are mutually exclusive. Therefore, only one of them can be specified at a time.
- format: the file format, which can only be CSV.
- encoding: the file data encoding format. It supports common PostgreSQL encoding formats, such as UTF-8.
- parse_errors: the fault-tolerant parsing mode. If an error occurs during the parsing process, the entire row of data is ignored.
- delimiter: the string used to delimit columns.
- quote: the quote character for files.
- escape: the escape character for files.

- null: sets the column matching the specified string to null. For example, null 'test' is used to set the value of the 'test' column to null.
- force_not_null: sets the value of a column to a non-null value. For example, force_not_null 'id' is used to set the value of the 'id' column to empty strings.
- compressiontype: the format of the files to be read or written in OSS.
  - none: The files are uncompressed. This is the default value.
  - gzip: The files are compressed in the GZIP format.
- compressionlevel: the degree to which data files written to OSS are compressed. Valid values: 1 to 9. Default value: 6.

> **Note**
> - You must specify filepath and dir in the OPTIONS parameter.
> - You must specify filepath or dir.
> - The export mode can only be dir.

Export mode parameters for CREATE FOREIGN TABLE

- oss_flush_block_size: the buffer size for the data written to OSS at a time. Default value: 32. Valid values: 1 to 128. Unit: MB.
- oss_file_max_size: the maximum size of a data file allowed to be written to OSS. If a data file reaches the maximum size, the remaining data is written to another data file. Default value: 1024. Valid values: 8 to 4000. Unit: MB.
- num_parallel_worker: the maximum number of threads that are allowed to run in parallel to compress the data written to OSS. Valid values: 1 to 8. Default value: 3.

Auxiliary functions

FUNCTION oss_fdw_list_file (relname text, schema text DEFAULT 'public')

- This function obtains the name and size of the OSS file that a foreign table matches.
- The file size is measured in bytes.

The following result is returned after `select * from oss_fdw_list_file('t_oss');` is executed:

```
              name              |   size
--------------------------------+-----------
 oss_test/test.gz.1  | 739698350
 oss_test/test.gz.2  | 739413041
 oss_test/test.gz.3  | 739562048
(3 rows)
```

Auxiliary features

oss_fdw.rds_read_one_file: In read mode, this feature is used to specify a file to match the foreign table. The foreign table matches only the specified file during data import.

Example: set oss_fdw.rds_read_one_file = 'oss_test/example16.csv.1';

The following result is returned after `set oss_fdw.rds_read_one_file = 'oss_test/test.gz.2';` and `select * from oss_fdw_list_file('t_oss');` are executed:

```
              name              |   size
--------------------------------+-----------
  oss_test/test.gz.2  | 739413041
(1 rows)
```

oss_fdw example

```
# Create the plug-in for a PostgreSQL database.
create extension oss_fdw; -- For a PPAS database, execute select rds_manage_extension('crea
te','oss_fdw');
# Create a server.
CREATE SERVER ossserver FOREIGN DATA WRAPPER oss_fdw OPTIONS
     (host 'oss-cn-hangzhou.aliyuncs.com', id 'xxx', key 'xxx', bucket 'mybucket');
# Create an OSS foreign table.
CREATE FOREIGN TABLE ossexample
    (date text, time text, open float,
     high float, low float, volume int)
     SERVER ossserver
     OPTIONS ( filepath 'osstest/example.csv', delimiter ',' ,
         format 'csv', encoding 'utf8', PARSE_ERRORS '100');
# Create a table named example to which to import data.
create table example
        (date text, time text, open float,
         high float, low float, volume int);
# Load data from ossexample to example.
insert into example select * from ossexample;
# Result
# oss_fdw estimates the file size in OSS and formulates a query plan.
explain insert into example select * from ossexample;
                             QUERY PLAN
--------------------------------------------------------------------
 Insert on example  (cost=0.00..1.60 rows=6 width=92)
   ->  Foreign Scan on ossexample  (cost=0.00..1.60 rows=6 width=92)
         Foreign OssFile: osstest/example.csv.0
         Foreign OssFile Size: 728
(4 rows)
# Write the data in the example table to OSS.
insert into ossexample select * from example;
explain insert into ossexample select * from example;
                           QUERY PLAN
-----------------------------------------------------------------
 Insert on ossexample  (cost=0.00..16.60 rows=660 width=92)
   ->  Seq Scan on example  (cost=0.00..16.60 rows=660 width=92)
(2 rows)
```

Additional considerations

- oss_fdw is a foreign table plug-in developed based on the PostgreSQL FOREIGN TABLE framework.

- The data import performance varies based on the PostgreSQL cluster resources (CPU, I/O, and memory) and OSS.
- To ensure data import performance, the ApsaraDB RDS for PostgreSQL instance must be in the same region as the OSS bucket.

ID and key encryption

If the id and key parameters for CREATE SERVER are not encrypted, the `select * from pg_foreign_server` statement execution result displays the information. Your AccessKey ID and AccessKey secret are exposed. You can use symmetric encryption to hide your AccessKey ID and AccessKey secret. Use different AccessKey pairs for different instances to further protect your information. However, to avoid incompatibility with earlier versions, do not add data types as you would in Greenplum.

Encrypted information:

```
postgres=# select * from pg_foreign_server ;
  srvname  | srvowner | srvfdw | srvtype | srvversion | srvacl |
srvoptions
-----------+----------+--------+---------+------------+--------+---------------------------
-------------------------------------------------------------------------------------------
--------------
---------------------------------
 ossserver |       10 |  16390 |         |            |        | {host=oss-cn-hangzhou-zmf.
aliyuncs.com,id=MD5xxxxxxxx,key=MD5xxxxxxxx,bucket=067862}
```

The encrypted information is preceded by the MD5 hash value. The remainder of the total length divided by 8 is 3. Therefore, encryption is not performed again when the exported data is imported. You cannot create an AccessKey pair that is preceded by MD5.

17.Use Pgpool for read/write splitting in ApsaraDB RDS for PostgreSQL

This topic describes how to use the Pgpool tool of PostgreSQL installed on an ECS instance to implement read/write splitting for your primary and read-only ApsaraDB RDS for PostgreSQL instances.

Context

If you do not use Pgpool to ensure high availability, Pgpool is stateless. The decrease in performance can be ignored. Additionally, Pgpool supports horizontal scaling of your database system. You can use Pgpool and the high availability architecture of ApsaraDB RDS for PostgreSQL to implement read/write splitting.

Set up a test environment

If you have purchased a primary ApsaraDB RDS instance that runs PostgreSQL 10 and have attached read-only instances to the primary instance, you need only to install Pgpool . For more information, see Create an instance and Create a read-only ApsaraDB RDS for PostgreSQL instance. After you install Pgpool, go to Configure Pgpool.

1. Run the `vi /etc/sysctl.conf` command to open the sysctl.conf file. Modify the following configurations:

```
# add by digoal.zhou
fs.aio-max-nr = 1048576
fs.file-max = 76724600
# Optional. Set the kernel.core_pattern parameter to /data01/corefiles/core_%e_%u_%t_%s
.%p.
# The /data01/corefiles directory that is used to store core dumps is created with the
777 permission before testing. If a symbolic link is used, change the directory to 777.

kernel.sem = 4096 2147483647 2147483646 512000
# Specify the semaphore. You can run the ipcs -l or -u command to obtain the semaphore
count. Each group of 16 processes requires a semaphore with a count of 17.

kernel.shmall = 107374182
# Specify the total size of shared memory segments. Recommended value: 80% of the memor
y capacity. Unit: pages.
kernel.shmmax = 274877906944
# Specify the maximum size of a single shared memory segment. Recommended value: 50% of
the memory capacity. Unit: bytes. In PostgreSQL versions later than 9.2, the use of sha
red memory significantly drops.
kernel.shmmni = 819200
# Specify the total number of shared memory segments that can be generated. At least tw
o shared memory segments must be generated within each PostgreSQL cluster.
net.core.netdev_max_backlog = 10000
net.core.rmem_default = 262144
# The default setting of the socket receive buffer in bytes.
net.core.rmem_max = 4194304
# The maximum receive socket buffer size in bytes
```

```
net.core.wmem_default = 262144
# The default setting (in bytes) of the socket send buffer.
net.core.wmem_max = 4194304
# The maximum send socket buffer size in bytes.
net.core.somaxconn = 4096
net.ipv4.tcp_max_syn_backlog = 4096
net.ipv4.tcp_keepalive_intvl = 20
net.ipv4.tcp_keepalive_probes = 3
net.ipv4.tcp_keepalive_time = 60
net.ipv4.tcp_mem = 8388608 12582912 16777216
net.ipv4.tcp_fin_timeout = 5
net.ipv4.tcp_synack_retries = 2
net.ipv4.tcp_syncookies = 1
# Enable SYN cookies. If an SYN waiting queue overflows, you can enable SYN cookies to
defend against a small number of SYN attacks.
net.ipv4.tcp_timestamps = 1
# Reduce the time after which a network socket enters the TIME-WAIT state.
net.ipv4.tcp_tw_recycle = 0
# If you set this parameter to 1 to enable the recycle function, network sockets in the
TIME-WAIT state over TCP connections are recycled. However, if network address translat
ion (NAT) is used, TCP connections may fail. We recommend that you set this parameter t
o 0 on the database server.
net.ipv4.tcp_tw_reuse = 1
# Enable the reuse function. This function enables network sockets in the TIME-WAIT sta
te to be reused over new TCP connections.
net.ipv4.tcp_max_tw_buckets = 262144
net.ipv4.tcp_rmem = 8192 87380 16777216
net.ipv4.tcp_wmem = 8192 65536 16777216
net.nf_conntrack_max = 1200000
net.netfilter.nf_conntrack_max = 1200000
vm.dirty_background_bytes = 409600000
# If the size of dirty pages reaches the specified limit, a background scheduling proce
ss (for example, pdflush) is invoked to flush the dirty pages to disks. These are the p
ages that are generated n seconds earlier. The value of n is calculated by using the fo
llowing formula: n = Value of the dirty_expire_centisecs parameter/100.
# The default limit is 10% of the memory capacity. If the memory capacity is large, we
recommend that you specify the limit in bytes.
vm.dirty_expire_centisecs = 3000
# Specify the maximum period to retain dirty pages. Dirty pages are flushed to disks af
ter the time period specified by this parameter elapses. The value 3000 indicates 30 se
conds.
vm.dirty_ratio = 95
# The processes that users call to write data onto disks must actively flush dirty page
s to disks. This applies when the background scheduling process to flush dirty pages is
slow and the size of dirty pages exceeds 95% of the memory capacity. These processes in
clude fsync and fdatasync.
# Set this parameter properly to prevent user-called processes from flushing dirty page
s to disks. This allows you to create multiple ApsaraDB RDS instances on a single serve
r and use control groups to limit the input/output operations per second (IOPS) per ins
tance.
vm.dirty_writeback_centisecs = 100
# Specify the time interval at which the background scheduling process (such as pdflush
) flushes dirty pages to disks. The value 100 indicates 1 second.
vm.swappiness = 0
# Disable the swap function
```

```
# Disable the swap function.
vm.mmap_min_addr = 65536
vm.overcommit_memory = 0
# Specify whether you can allocate more memory space than the physical host has availab
le. If you set this parameter to 1, the system always considers the available memory sp
ace sufficient. If the memory capacity provided in the test environment is low, we reco
mmend that you set this parameter to 1.
vm.overcommit_ratio = 90
# Specify the memory capacity that can be allocated when the overcommit_memory paramete
r is set to 2.
vm.swappiness = 0
# Disable the swap function.
vm.zone_reclaim_mode = 0
# Disable non-uniform memory access (NUMA). You can also disable NUMA in the vmlinux fi
le.
net.ipv4.ip_local_port_range = 40000 65535
# Specify the range of TCP or UDP port numbers for the physical host to allocate.

fs.nr_open=20480000
# Specify the maximum number of file handles that a single process can open.

# Take note of the following parameters:
#vm.extra_free_kbytes = 4096000   # If the physical host provides a low memory capacity
, do not specify a large value such as 4096000. If you specify a large value, the physi
cal host may not start.
#vm.min_free_kbytes = 6291456    # We recommend that you increase the value of the vm.m
in_free_kbytes parameter by 1 GB for every 32 GB of memory.
# If the physical host does not provide much memory, we recommend that you do not confi
gure vm.extra_free_kbytes and vm.min_free_kbytes.
# vm.nr_hugepages = 66536
# If the size of the shared buffer exceeds 64 GB, we recommend that you use huge pages.
You can specify the page size by setting the Hugepagesize parameter in the /proc/meminf
o file.
#vm.lowmem_reserve_ratio = 1 1 1
# If the memory capacity exceeds 64 GB, we recommend that you set this parameter. Other
wise, we recommend that you retain the default value 256 256 32.
```

2. Run the `vi /etc/security/limits.conf` command to open the limits.conf file. Modify the following configurations:

```
* soft    nofile  1024000
* hard    nofile  1024000
* soft    nproc   unlimited
* hard    nproc   unlimited
* soft    core    unlimited
* hard    core    unlimited
* soft    memlock unlimited
* hard    memlock unlimited
# Comment out the other parameters in the limits.conf file.
# Comment out the /etc/security/limits.d/20-nproc.conf file.
```

3. Run the following commands to open the rc.local file:

```
chmod +x /etc/rc.local
vi /etc/rc.local
```

Modify the following configurations to disable transparent huge pages, configure huge pages, and start PostgreSQL:

```
# Disable transparent huge pages.
if test -f /sys/kernel/mm/transparent_hugepage/enabled; then
   echo never > /sys/kernel/mm/transparent_hugepage/enabled
fi
# Configure huge pages for two instances. Each instance has a shared buffer of 16 GB.
sysctl -w vm.nr_hugepages=17000
# Start the two instances.
su - postgres -c "pg_ctl start -D /data01/pg12_3389/pg_root"
su - postgres -c "pg_ctl start -D /data01/pg12_8002/pg_root"
```

4. Create a file system.

> **Warning** If you use a new disk, you must verify that the new disk belongs to the vdb partition instead of the vda partition. If the new disk belongs to the vda partition, data may be deleted from the new disk.

```
parted -a optimal -s /dev/vdb mklabel gpt mkpart primary 1MiB 100%FREE
mkfs.ext4 /dev/vdb1 -m 0 -O extent,uninit_bg -E lazy_itable_init=1 -b 4096 -T largefile
-L vdb1
vi /etc/fstab
LABEL=vdb1 /data01 ext4 defaults,noatime,nodiratime,nodelalloc,barrier=0,data=writeback
0 0
mkdir /data01
mount -a
```

5. Start the irqbalance command line tool.

```
systemctl status irqbalance
systemctl enable irqbalance
systemctl start irqbalance
systemctl status irqbalance
```

6. Install PostgreSQL 10 and Pgpool.

```
yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

yum install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg-
redhat-repo-latest.noarch.rpm
yum search all postgresql
yum search all pgpool
yum install -y postgresql12*
yum install -y pgpool-II-12-extensions
```

7. Initialize the data directory of your database system.

```
mkdir /data01/pg12_3389
chown postgres:postgres /data01/pg12_3389
```

8. Configure environment variables for the postgres user.

```
su - postgres
vi .bash_profile
```

Append the following parameters to the environment variables:

```
export PS1="$USER@`/bin/hostname -s`-> "
export PGPORT=3389
export PGDATA=/data01/pg12_$PGPORT/pg_root
export LANG=en_US.utf8
export PGHOME=/usr/pgsql-12
export LD_LIBRARY_PATH=$PGHOME/lib:/lib64:/usr/lib64:/usr/local/lib64:/lib:/usr/lib:/usr/local/lib:$LD_LIBRARY_PATH
export DATE=`date +"%Y%m%d%H%M"`
export PATH=$PGHOME/bin:$PATH:.
export MANPATH=$PGHOME/share/man:$MANPATH
export PGHOST=$PGDATA
export PGUSER=postgres
export PGDATABASE=db1
alias rm='rm -i'
alias ll='ls -lh'
unalias vi
```

9. Initialize your primary ApsaraDB RDS instance.

```
initdb -D $PGDATA -U postgres -E UTF8 --lc-collate=C --lc-ctype=en_US.utf8
```

10. Modify the postgresql.conf file.

```
listen_addresses = '0.0.0.0'
port = 3389
max_connections = 1500
superuser_reserved_connections = 13
unix_socket_directories = '., /var/run/postgresql, /tmp'
tcp_keepalives_idle = 60
tcp_keepalives_interval = 10
tcp_keepalives_count = 10
shared_buffers = 16GB
huge_pages = on
work_mem = 8MB
maintenance_work_mem = 1GB
dynamic_shared_memory_type = posix
vacuum_cost_delay = 0
bgwriter_delay = 10ms
bgwriter_lru_maxpages = 1000
bgwriter_lru_multiplier = 10.0
bgwriter_flush_after = 512kB
effective_io_concurrency = 0
max_worker_processes = 128
max_parallel_maintenance_workers = 3
max_parallel_workers_per_gather = 4
parallel_leader_participation = off
max_parallel_workers = 8
backend_flush_after = 256
wal_level = replica
synchronous_commit = off
```

```
full_page_writes = on
wal_compression = on
wal_buffers = 16MB
wal_writer_delay = 10ms
wal_writer_flush_after = 1MB
checkpoint_timeout = 15min
max_wal_size = 64GB
min_wal_size = 8GB
checkpoint_completion_target = 0.2
checkpoint_flush_after = 256kB
random_page_cost = 1.1
effective_cache_size = 48GB
log_destination = 'csvlog'
logging_collector = on
log_directory = 'log'
log_filename = 'postgresql-%a.log'
log_truncate_on_rotation = on
log_rotation_age = 1d
log_rotation_size = 0
log_min_duration_statement = 1s
log_checkpoints = on
log_connections = on
log_disconnections = on
log_line_prefix = '%m [%p] '
log_statement = 'ddl'
log_timezone = 'Asia/Shanghai'
autovacuum = on
log_autovacuum_min_duration = 0
autovacuum_vacuum_scale_factor = 0.1
autovacuum_analyze_scale_factor = 0.05
autovacuum_freeze_max_age = 800000000
autovacuum_multixact_freeze_max_age = 900000000
autovacuum_vacuum_cost_delay = 0
vacuum_freeze_table_age = 750000000
vacuum_multixact_freeze_table_age = 750000000
datestyle = 'iso, mdy'
timezone = 'Asia/Shanghai'
lc_messages = 'en_US.utf8'
lc_monetary = 'en_US.utf8'
lc_numeric = 'en_US.utf8'
lc_time = 'en_US.utf8'
default_text_search_config = 'pg_catalog.english'
```

11. Modify the pg_hba.conf file.

> **Note** Pgpool-II is installed on the same ECS instance as the database server where PostgreSQL resides. If you specify the 127.0.0.1 IP address in the pg_hba.conf file, you must enter the correct password to ensure a successful logon.

```
# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     trust
host    replication     all             127.0.0.1/32            trust
host    replication     all             ::1/128                 trust
host db123 digoal 0.0.0.0/0 md5
```

12. Execute a statement in the database to create a user authorized with streaming replication permissions. Example:

```
create role rep123 login replication encrypted password 'xxxxxxx';
```

13. Execute statements in the database to create a user and authorize it to manage your ApsaraDB RDS instances. Example:

```
create role digoal login encrypted password 'xxxxxxx';
create database db123 owner digoal;
```

14. Create a user who is authorized to check the health heartbeats between Pgpool and your read-only ApsaraDB RDS instances. With the parameters of Pgpool properly configured, this user can check the write-ahead logging (WAL) replay latency on each read-only ApsaraDB RDS instance. Example:

```
create role nobody login encrypted password 'xxxxxxx';
```

Create a secondary ApsaraDB RDS instance

To simplify the test procedure, perform the following steps to create a secondary ApsaraDB RDS instance on the same ECS instance as your primary ApsaraDB RDS instance:

1. Use the pg_basebackup tool to create a secondary ApsaraDB RDS instance.

```
pg_basebackup -D /data01/pg12_8002/pg_root -F p --checkpoint=fast -P -h 127.0.0.1 -p 3389 -U rep123
```

2. Run the following commands to open the postgresql.conf file of the secondary ApsaraDB RDS instance:

```
cd /data01/pg12_8002/pg_root
vi postgresql.conf
```

Modify the following configurations:

```
# The secondary ApsaraDB RDS instance has the following configurations different from t
he primary ApsaraDB RDS instance:
port = 8002
primary_conninfo = 'hostaddr=127.0.0.1 port=3389 user=rep123' # You do not need to set
the password. This is because trust relationships are configured on the primary ApsaraD
B RDS instance.
hot_standby = on
wal_receiver_status_interval = 1s
wal_receiver_timeout = 10s
recovery_target_timeline = 'latest'
```

3. Configure the standby.signal file of the secondary ApsaraDB RDS instance.

```
cd /data01/pg12_8002/pg_root
touch standby.signal
```

4. Execute the `SELECT * FROM pg_stat_replication ;` statement in the database to check whether data is properly synchronized between the primary and secondary ApsaraDB RDS instances. The following output is returned:

```
-[ RECORD 1 ]----+------------------------------
pid              | 21065
usesysid         | 10
usename          | postgres
application_name | walreceiver
client_addr      | 127.0.0.1
client_hostname  |
client_port      | 47064
backend_start    | 2020-02-29 00:26:28.485427+08
backend_xmin     |
state            | streaming
sent_lsn         | 0/52000060
write_lsn        | 0/52000060
flush_lsn        | 0/52000060
replay_lsn       | 0/52000060
write_lag        |
flush_lag        |
replay_lag       |
sync_priority    | 0
sync_state       | async
reply_time       | 2020-02-29 01:32:40.635183+08
```

Configure Pgpool

1. Query the location where Pgpool is installed.

```
rpm -qa|grep pgpool
pgpool-II-12-extensions-4.1.1-1.rhel7.x86_64
pgpool-II-12-4.1.1-1.rhel7.x86_64
rpm -ql pgpool-II-12-4.1.1
```

2. Run the following commands to open the pgpool.conf file:

```
cd /etc/pgpool-II-12/
cp pgpool.conf.sample-stream pgpool.conf
vi pgpool.conf
```

Modify the following configurations:

```
listen_addresses = '0.0.0.0'
port = 8001
socket_dir = '/tmp'
reserved_connections = 0
pcp_listen_addresses = ''
pcp_port = 9898
pcp_socket_dir = '/tmp'
# - Backend Connection Settings -
backend_hostname0 = '127.0.0.1'
                                    # Host name or IP address to connect to for backend
0
backend_port0 = 3389
                                    # Port number for backend 0
backend_weight0 = 1
                                    # Weight for backend 0 (only in load balancing mode)

backend_data_directory0 = '/data01/pg12_3389/pg_root'
                                    # Data directory for backend 0
backend_flag0 = 'ALWAYS_MASTER'
                                    # Controls various backend behavior
                                    # ALLOW_TO_FAILOVER, DISALLOW_TO_FAILOVER
                                    # or ALWAYS_MASTER
backend_application_name0 = 'server0'
                                    # walsender's application_name, used for "show pool_
nodes" command
backend_hostname1 = '127.0.0.1'
backend_port1 = 8002
backend_weight1 = 1
backend_data_directory1 = '/data01/pg12_8002/pg_root'
backend_flag1 = 'DISALLOW_TO_FAILOVER'
backend_application_name1 = 'server1'
# - Authentication -
enable_pool_hba = on
                                    # Use pool_hba.conf for client authentication
pool_passwd = 'pool_passwd'
                                    # File name of pool_passwd for md5 authentication.
                                    # "" disables pool_passwd.
                                    # (change requires restart)
allow_clear_text_frontend_auth = off
                                    # Allow Pgpool-II to use clear text password authent
ication
                                    # with clients, when pool_passwd does not
                                    # contain the user password
# - Concurrent session and pool size -
num_init_children = 128
                                    # Number of concurrent sessions allowed
                                    # (change requires restart)
max_pool = 4
```

```
                                    # Number of connection pool caches per connection
                                    # (change requires restart)
# - Life time -
child_life_time = 300
                                    # Pool exits after being idle for this many seconds

child_max_connections = 0
                                    # Pool exits after receiving that many connections
                                    # 0 means no exit
connection_life_time = 0
                                    # Connection to backend closes after being idle for
this many seconds
                                    # 0 means no close
client_idle_limit = 0
                                    # Client is disconnected after being idle for that m
any seconds
                                    # (even inside an explicit transactions!)
                                    # 0 means no disconnection
#------------------------------------------------------------------------------
# LOGS
#------------------------------------------------------------------------------
# - Where to log -
log_destination = 'syslog'
                                    # Where to log
                                    # Valid values are combinations of stderr,
                                    # and syslog. Default to stderr.
log_connections = on
                                    # Log connections
log_standby_delay = 'if_over_threshold'
                                    # Log standby delay
                                    # Valid values are combinations of always,
                                    # if_over_threshold, none
#------------------------------------------------------------------------------
# FILE LOCATIONS
#------------------------------------------------------------------------------
pid_file_name = '/var/run/pgpool-II-12/pgpool.pid'
                                    # PID file name
                                    # Can be specified as relative to the"
                                    # location of pgpool.conf file or
                                    # as an absolute path
                                    # (change requires restart)
logdir = '/tmp'
                                    # Directory of pgPool status file
                                    # (change requires restart)
#------------------------------------------------------------------------------
# CONNECTION POOLING
#------------------------------------------------------------------------------
connection_cache = on
                                    # Activate connection pools
                                    # (change requires restart)
                                    # Semicolon separated list of queries
                                    # to be issued at the end of a session
                                    # The default is for 8.3 and later
reset_query_list = 'ABORT; DISCARD ALL'
```

```
#------------------------------------------------------------------------------
# LOAD BALANCING MODE
#------------------------------------------------------------------------------
load_balance_mode = on
                                    # Activate load balancing mode
                                    # (change requires restart)
ignore_leading_white_space = on
                                    # Ignore leading white spaces of each query
white_function_list = ''
                                    # Comma separated list of function names
                                    # that don't write to database
                                    # Regexp are accepted
black_function_list = 'currval,lastval,nextval,setval'
                                    # Comma separated list of function names
                                    # that write to database
                                    # Regexp are accepted
black_query_pattern_list = ''
                                    # Semicolon separated list of query patterns
                                    # that should be sent to primary node
                                    # Regexp are accepted
                                    # valid for streaming replicaton mode only.
database_redirect_preference_list = ''
                                    # comma separated list of pairs of database and node
id.
                                    # example: postgres:primary,mydb[0-4]:1,mydb[5-9]:2'

                                    # valid for streaming replicaton mode only.
app_name_redirect_preference_list = ''
                                    # comma separated list of pairs of app name and node
id.
                                    # example: 'psql:primary,myapp[0-4]:1,myapp[5-9]:sta
ndby'
                                    # valid for streaming replicaton mode only.
allow_sql_comments = off
                                    # if on, ignore SQL comments when judging if load ba
lance or
                                    # query cache is possible.
                                    # If off, SQL comments effectively prevent the judgm
ent
                                    # (pre 3.4 behavior).
disable_load_balance_on_write = 'transaction'
                                    # Load balance behavior when write query is issued
                                    # in an explicit transaction.
                                    # Note that any query not in an explicit transaction

                                    # is not affected by the parameter.
                                    # 'transaction' (the default): if a write query is i
ssued,
                                    # subsequent read queries will not be load balanced

                                    # until the transaction ends.
                                    # 'trans_transaction': if a write query is issued,
                                    # subsequent read queries in an explicit transaction

                                    # will not be load balanced until the session ends
```

```
                                    # will not be load balanced until the session ends.

                                    # 'always': if a write query is issued, read queries
will
                                    # not be load balanced until the session ends.
statement_level_load_balance = off
                                    # Enables statement level load balancing
#------------------------------------------------------------------------------
# MASTER/SLAVE MODE
#------------------------------------------------------------------------------
master_slave_mode = on
                                    # Activate master/slave mode
                                    # (change requires restart)
master_slave_sub_mode = 'stream'
                                    # Master/slave sub mode
                                    # Valid values are combinations stream, slony
                                    # or logical. Default is stream.
                                    # (change requires restart)
# - Streaming -
sr_check_period = 3
                                    # Streaming replication check period
                                    # Disabled (0) by default
sr_check_user = 'nobody'
                                    # Streaming replication check user
                                    # This is neccessary even if you disable streaming
                                    # replication delay check by sr_check_period = 0
sr_check_password = ''
                                    # Password for streaming replication check user
                                    # Leaving it empty will make Pgpool-II to first look
for the
                                    # Password in pool_passwd file before using the empt
y password
sr_check_database = 'postgres'
                                    # Database name for streaming replication check
delay_threshold = 512000
                                    # Threshold before not dispatching query to standby
node
                                    # Unit is in bytes
                                    # Disabled (0) by default
#------------------------------------------------------------------------------
# HEALTH CHECK GLOBAL PARAMETERS
#------------------------------------------------------------------------------
health_check_period = 5
                                    # Health check period
                                    # Disabled (0) by default
health_check_timeout = 10
                                    # Health check timeout
                                    # 0 means no timeout
health_check_user = 'nobody'
                                    # Health check user
health_check_password = ''
                                    # Password for health check user
                                    # Leaving it empty will make Pgpool-II to first look
for the
                                    # Password in pool_passwd file before using the empt
```

```
                                    # password in pool_passwd file before using the expe
y password
health_check_database = ''
                                    # Database name for health check. If '', tries 'post
gres' frist,
health_check_max_retries = 60
                                    # Maximum number of times to retry a failed health c
heck before giving up.
health_check_retry_delay = 1
                                    # Amount of time to wait (in seconds) between retrie
s.
connect_timeout = 10000
                                    # Timeout value in milliseconds before giving up to
connect to backend.
                                    # Default is 10000 ms (10 second). Flaky network use
r may want to increase
                                    # the value. 0 means no timeout.
                                    # Note that this value is not only used for health c
heck,
                                    # but also for ordinary conection to backend.
#------------------------------------------------------------------------------
# FAILOVER AND FAILBACK
#------------------------------------------------------------------------------
failover_on_backend_error = off
                                    # Initiates failover when reading/writing to the
                                    # backend communication socket fails
                                    # If set to off, pgpool will report an
                                    # error and disconnect the session.
relcache_expire = 0  # After the configuration file is restructured, we recommend that
you set this parameter to 1, reload the configuration file, and then set this parameter
to 0 again. You can also set this parameter to a specific point in time.
                                    # Life time of relation cache in seconds.
                                    # 0 means no cache expiration(the default).
                                    # The relation cache is used for cache the
                                    # query result against PostgreSQL system
                                    # catalog to obtain various information
                                    # including table structures or if it's a
                                    # temporary table or not. The cache is
                                    # maintained in a pgpool child local memory
                                    # and being kept as long as it survives.
                                    # If someone modify the table by using
                                    # ALTER TABLE or some such, the relcache is
                                    # not consistent anymore.
                                    # For this purpose, cache_expiration
                                    # controls the life time of the cache.
relcache_size = 8192
                                    # Number of relation cache
                                    # entry. If you see frequently:
                                    # "pool_search_relcache: cache replacement happend"

                                    # in the pgpool log, you might want to increate this
number.
```

3. Run the `cd /etc/pgpool-II-12` command to configure the pool_passwd file.

> **Note** If you connect to your ApsaraDB RDS instances by using Pgpool, you must configure the pool_passwd file. This is because Pgpool supports the authentication protocol of PostgreSQL.

```
# Run the following command:
#pg_md5 --md5auth --username=username password
# Generate the passwords of the digoal and nobody users. The passwords are automaticall
y written into the pool_passwd file.
pg_md5 --md5auth --username=digoal "xxxxxxx"
pg_md5 --md5auth --username=nobody "xxxxxxx"
```

4. Use the system to automatically generate the pool_passwd file.

```
cd /etc/pgpool-II-12
cat pool_passwd
```

5. Run the following commands to configure the pgpool_hba file:

```
cd /etc/pgpool-II-12
cp pool_hba.conf.sample pool_hba.conf
vi pool_hba.conf
```

Configure the following parameters:

```
host all all 0.0.0.0/0 md5
```

6. Configure the pcp.conf file.

> **Note** The pcp.conf file is used to manage the users and passwords of Pgpool. It is not related to the users and passwords of your ApsaraDB RDS instances.

```
cd /etc/pgpool-II-12
# pg_md5 abc  # In this command, you set the password to abc and encrypt it by using th
e MD5 encryption algorithm.
900150983cd24fb0d6963f7d28e17f72
cp pcp.conf.sample pcp.conf
vi pcp.conf
# USERID:MD5PASSWD
manage:900150983cd24fb0d6963f7d28e17f72  # In this command, the manage user is used to
manage PCP.
```

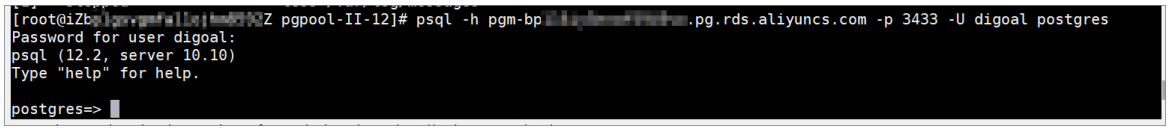
7. Start Pgpool.

```
cd /etc/pgpool-II-12
pgpool -f ./pgpool.conf -a ./pool_hba.conf -F ./pcp.conf
```

> **Note** If you want to view the logs of Pgpool, run the following command:
>
> ```
> less /var/log/messages
> ```

8. Use Pgpool to connect to your ApsaraDB RDS instances.

```
psql -h 127.0.0.1 -p 8001 -U digoal postgres
```



FAQ

- Q: How do I test whether read/write splitting is enabled?

  A: You can connect to your ApsaraDB RDS instances by using Pgpool and call the pg_is_in_recovery() function. Then, close the connection, establish a connection again, and call the pg_is_in_recovery() function again. If you receive a value of false and then a value of true, Pgpool routes requests to your primary ApsaraDB RDS instance and then to your read-only ApsaraDB RDS instances, and read/write splitting is enabled.

- Q: Does Pgpool increase the latency?

  A: Pgpool increases the latency slightly. In the test environment you set up in this topic, the latency increases by about 0.12 milliseconds.

- Q: How does Pgpool check the latency and health on my read-only ApsaraDB RDS instances?
  - A: If the WAL replay latency on a read-only ApsaraDB RDS instance exceeds the specified limit, Pgpool stops routing SQL requests to the read-only instance. Pgpool resumes routing SQL requests to the read-only instance only after it detects that the WAL replay latency on the read-only instance falls below the specified limit.

    > **Note** Connect to your primary ApsaraDB RDS instance and query the location where the current WAL data record is written. This location is referred to as log sequence number (LSN) 1. Then, connect to a read-only ApsaraDB RDS instance and query the location where the current WAL data record is replayed. This location is referred to as LSN 2. You can obtain the number of bytes between LSN 1 and LSN 2. This number indicates the latency.

  - Pgpool monitors the health of your read-only ApsaraDB RDS instances. If a read-only instance is unhealthy, Pgpool stops routing requests to the read-only instance.

- Q: How do I stop Pgpool and reload the configuration of Pgpool?

  A: Run the `pgpool --help` command to obtain more information about the commands used in Pgpool. Example:

  ```
  cd /etc/pgpool-II-12
  pgpool -f ./pgpool.conf -m fast stop
  ```

- Q: How do I configure Pgpool if more than one read-only ApsaraDB RDS instance is attached to my primary ApsaraDB RDS instance?

  A: Add the configurations of all the attached read-only ApsaraDB RDS instances to the pgpool.conf file. Example:

```
backend_hostname1 = 'xx.xx.xxx.xx'
backend_port1 = 8002
backend_weight1 = 1
backend_data_directory1 = '/data01/pg12_8002/pg_root'
backend_flag1 = 'DISALLOW_TO_FAILOVER'
backend_application_name1 = 'server1'
backend_hostname2 = 'xx.xx.xx.xx'
backend_port1 = 8002
backend_weight1 = 1
backend_data_directory1 = '/data01/pg12_8002/pg_root'
backend_flag1 = 'DISALLOW_TO_FAILOVER'
backend_application_name1 = 'server1'
```

- Q: How do I use pcp commands to view the status of my read-only ApsaraDB RDS instances?

  A: To obtain the status of your read-only ApsaraDB RDS instances by using pcp commands, run the following command:

```
# pcp_node_info -U manage -h /tmp -p 9898 -n 1 -v
Password: # Enter the password.
Hostname               : 127.0.0.1
Port                   : 8002
Status                 : 2
Weight                 : 0.500000
Status Name            : up
Role                   : standby
Replication Delay      : 0
Replication State      :
Replication Sync State :
Last Status Change     : 2020-02-29 00:20:29
```

- Q: Which listening ports are used by Pgpool for read/write splitting?

  A: The following listening ports are used by Pgpool for read/write splitting:

  - Primary ApsaraDB RDS instance: Port 3389
  - Secondary ApsaraDB RDS instance: Port 8002
  - Pgpool: Port 8001
  - PCP: Port 9898

18.Use ShardingSphere to develop ApsaraDB RDS for PostgreSQL

ShardingSphere is an open source ecosystem that consists of a set of distributed database middleware solutions.

Prerequisites

All PostgreSQL versions used with ApsaraDB RDS support ShardingSphere.

Context

ApsaraDB RDS for PostgreSQL supports database-integrated sharding plug-ins (such as Citus, Postgres-XC, and AntDB) and massively parallel processing (MPP) products. It also supports sharding middleware products that are similar to those widely used in MySQL, such as ShardingSphere.

ShardingSphere is suitable for services that run in databases with thorough, well-organized logical sharding. It offers the following features:

- Data sharding
  - Database and table sharding
  - Read/write splitting
  - Sharding strategy customization
  - Decentralized distributed primary key
- Distributed transaction
  - Unified transaction API
  - XA transaction
  - BASE transaction
- Database orchestration
  - Dynamic configuration
  - Orchestration and governance
  - Data encryption
  - Tracing and observability
  - Elastic scaling out (planning)

For more information, visit the ShardingSphere documentation.

ShardingSphere products

ShardingSphere includes three independent products. You can choose the product that best suits your business requirements. The following table describes these products.

Parameter	Sharding-JDBC	Sharding-Proxy	Sharding-Sidecar
Supported database engine	All JDBC-compatible database engines such as MySQL, PostgreSQL, Oracle, and SQL Server	MySQL and PostgreSQL	MySQL and PostgreSQL
Connections consumed	High	Low	High
Supported heterogeneous language	Java	All	All
Performance	Low consumption	Moderate consumption	Low consumption
Decentralized	Yes	No	Yes
Stateless API	No	Yes	No

Prepare configuration templates

1. On your ECS instance, run the following command to go to the directory where configuration templates are stored. The directory is under the root directory in this example.

```
cd /root/apache-shardingsphere-incubating-4.0.0-sharding-proxy-bin/conf
```

2. Run the `ll` command to view all files stored in the directory: Command output:

```
total 24
-rw-r--r-- 1 501 games 3019 Jul 30  2019 config-encrypt.yaml
-rw-r--r-- 1 501 games 3582 Apr 22  2019 config-master_slave.yaml
-rw-r--r-- 1 501 games 4278 Apr 22  2019 config-sharding.yaml
-rw-r--r-- 1 501 games 1918 Jul 30  2019 server.yaml
```

> **Note**
> - config-encrypt.yaml: the data encryption configuration file.
> - config-master_slave.yaml: the read/write splitting configuration file.
> - config-sharding.yaml: the data sharding configuration file.
> - server.yaml: the common configuration file.

3. Modify the configuration files.

> **Note** For more information about the configuration files, visit the ShardingSphere documentation. In this example, the data sharding and common configuration files are used.
> - Example of a data sharding configuration file:

```
schemaName: # The name of the logical data source.
dataSources: # The configuration of the data source. You can configure more than one data source by using the data_source_name variable.
  <data_source_name>: # You do not need to configure a database connection pool. This is different in Sharding-JDBC.
    url: # The URL used to connect to your database.
    username: # The username used to log on to the database.
    password: # The password used to log on to the database.
    connectionTimeoutMilliseconds: 30000 # The connection timeout period in milliseconds.
    idleTimeoutMilliseconds: 60000 # The idle-connection reclaiming timeout period in milliseconds.
    maxLifetimeMilliseconds: 1800000 # The maximum connection time to live (TTL) in milliseconds.
    maxPoolSize: 65 # The maximum number of connections allowed.
shardingRule: # You do not need to configure a sharding rule, because it is the same in Sharding-JDBC.
```

- Example of a common configuration file:

```
Proxy properties
# You do not need to configure proxy properties that are the same in Sharding-JDBC
props:
  acceptor.size: # The number of worker threads that receive requests from the client. The default number is equal to the number of CPU cores multiplied by 2.
  proxy.transaction.type: # The type of transaction processed by the proxy. Valid values: LOCAL | XA | BASE. Default value: LOCAL. Value XA specifies to use Atomikos as the transaction manager. Value BASE specifies to copy the .jar package that implements the ShardingTransactionManager operation to the lib directory.
  proxy.opentracing.enabled: # Specifies whether to enable link tracing. Link tracing is disabled by default.
  check.table.metadata.enabled: # Specifies whether to check the consistency of metadata among sharding tables during startup. Default value: false.
  proxy.frontend.flush.threshold: # The number of packets returned in a batch during a complex query.
Permission verification
This part of the configuration is used to verify your permissions when you attempt to log on to Sharding-Proxy. After you configure the username, password, and authorized databases, you must use the correct username and password to log on to Sharding-Proxy from the authorized databases.
authentication:
  users:
    root: # The username of the root user.
      password: root# The password of the root user.
    sharding: # The username of the sharding user.
      password: sharding# The password of the sharding user.
      authorizedSchemas: sharding_db, masterslave_db # The databases in which the specified user is authorized. If you want to specify more than one database, separate them with commas (,). You are granted the permissions of the root user by default. This way, you can access all databases.
```

Set up a test environment

- On your ECS instance, install Java.

```
yum install -y java
```

- Configure an ApsaraDB RDS instance that runs PostgreSQL 10.
  - Create an account with username r1.
  - Set the password of the account to "PW123321!".
  - Create the following databases whose owners are user r1: db0, db1, db2, and db3.
  - Add the IP address of your ECS instance to an IP address whitelist of the ApsaraDB RDS for PostgreSQL instance.

> **Note**
> - For more information about how to create an ApsaraDB RDS for PostgreSQL instance, database, and account, see Create an instance and Create a database and an account.
> - For more information about how to configure an IP address whitelist, see Configure an IP address whitelist.

- Run `vi /root/apache-shardingsphere-incubating-4.0.0-sharding-proxy-bin/conf/server.yaml` to configure the following common configuration file:

```
  authentication:
  users:
    r1:
      password: PW123321!
      authorizedSchemas: db0,db1,db2,db3
props:
  executor.size: 16
  sql.show: false
```

Test horizontal sharding

1. Run `vi /root/apache-shardingsphere-incubating-4.0.0-sharding-proxy-bin/conf/config-sharding.yaml` to modify the following data sharding configuration file:

```
schemaName: sdb
dataSources:
  db0:
    url: jdbc:postgresql://pgm-bpxxxxx.pg.rds.aliyuncs.com:1433/db0
    username: r1
    password: PW123321!
    connectionTimeoutMilliseconds: 30000
    idleTimeoutMilliseconds: 60000
    maxLifetimeMilliseconds: 1800000
    maxPoolSize: 65
  db1:
    url: jdbc:postgresql://pgm-bpxxxxx.pg.rds.aliyuncs.com:1433/db1
    username: r1
    password: PW123321!
    connectionTimeoutMilliseconds: 30000
```

```
    idleTimeoutMilliseconds: 60000
    maxLifetimeMilliseconds: 1800000
    maxPoolSize: 65
  db2:
    url: jdbc:postgresql://pgm-bpxxxxx.pg.rds.aliyuncs.com:1433/db2
    username: r1
    password: PW123321!
    connectionTimeoutMilliseconds: 30000
    idleTimeoutMilliseconds: 60000
    maxLifetimeMilliseconds: 1800000
    maxPoolSize: 65
  db3:
    url: jdbc:postgresql://pgm-bpxxxxx.pg.rds.aliyuncs.com:1433/db3
    username: r1
    password: PW123321!
    connectionTimeoutMilliseconds: 30000
    idleTimeoutMilliseconds: 60000
    maxLifetimeMilliseconds: 1800000
    maxPoolSize: 65
shardingRule:
  tables:
    t_order:
      actualDataNodes: db${0..3}.t_order${0..7}
      databaseStrategy:
        inline:
          shardingColumn: user_id
          algorithmExpression: db${user_id % 4}
      tableStrategy:
        inline:
          shardingColumn: order_id
          algorithmExpression: t_order${order_id % 8}
      keyGenerator:
        type: SNOWFLAKE
        column: order_id
    t_order_item:
      actualDataNodes: db${0..3}.t_order_item${0..7}
      databaseStrategy:
        inline:
          shardingColumn: user_id
          algorithmExpression: db${user_id % 4}
      tableStrategy:
        inline:
          shardingColumn: order_id
          algorithmExpression: t_order_item${order_id % 8}
      keyGenerator:
        type: SNOWFLAKE
        column: order_item_id
  bindingTables:
    - t_order,t_order_item
  defaultTableStrategy:
    none:
```

2. Start ShardingSphere and listen to Port 8001.

```
cd /root/apache-shardingsphere-incubating-4.0.0-sharding-proxy-bin/bin/
./start.sh 8001
```

3. Connect to the destination database.

```
psql -h 127.0.0.1 -p 8001 -U r1 sdb
```

4. Create a table.

```
create table t_order(order_id int8 primary key, user_id int8, info text, c1 int, crt_time timestamp);
create table t_order_item(order_item_id int8 primary key, order_id int8, user_id int8, info text, c1 int, c2 int, c3 int, c4 int, c5 int, crt_time timestamp);
```

> **Note** When you create a table, the system creates horizontal shards in the destination database based on the sharding strategy that you specify.

FAQ

- If you want to view the SQL parsing and routing statements used in ShardingSphere, run `vi /root/apache-shardingsphere-incubating-4.0.0-sharding-proxy-bin/conf/server.yaml`.

```
  authentication:
  users:
    r1:
      password: PW123321!
      authorizedSchemas: db0,db1,db2,db3
props:
  executor.size: 16
  sql.show: true # Specifies to log parsed SQL statements.
```

- If you want to test writes and queries, run the following commands:

```
insert into t_order (user_id, info, c1, crt_time) values (0,'a',1,now());
insert into t_order (user_id, info, c1, crt_time) values (1,'b',2,now());
insert into t_order (user_id, info, c1, crt_time) values (2,'c',3,now());
insert into t_order (user_id, info, c1, crt_time) values (3,'c',4,now());
select * from t_order;
```

The following result is returned in this example:

```
      order_id      | user_id | info | c1 |          crt_time
--------------------+---------+------+----+----------------------------
 433352561047633921 |       0 | a    |  1 | 2020-02-09 19:48:21.856555
 433352585668198400 |       1 | b    |  2 | 2020-02-09 19:48:27.726815
 433352610813050881 |       2 | c    |  3 | 2020-02-09 19:48:33.721754
 433352628370407424 |       3 | c    |  4 | 2020-02-09 19:48:37.907683
(4 rows)
```

- If you want to view ShardingSphere logs, run the following command:

```
/root/apache-shardingsphere-incubating-4.0.0-sharding-proxy-bin/logs/stdout.log
```

- If you want to use pgbench for stress testing, run the following commands:

```
vi test.sql
\set user_id random(1,100000000)
\set order_id random(1,2000000000)
\set order_item_id random(1,2000000000)
insert into t_order (user_id, order_id, info, c1 , crt_time) values (:user_id, :order_id,
random()::text, random()*1000, now()) on conflict (order_id) do update set info=excluded.
info,c1=excluded.c1,crt_time=excluded.crt_time;
insert into t_order_item (order_item_id, user_id, order_id, info, c1,c2,c3,c4,c5,crt_time
) values (:order_item_id, :user_id,:order_id,random()::text, random()*1000,random()*1000,
random()*1000,random()*1000,random()*1000, now()) on conflict(order_item_id) do nothing;

pgbench -M simple -n -r -P 1 -f ./test.sql -c 24 -j 24  -h 127.0.0.1 -p 8001 -U r1 sdb -T
120
progress: 1.0 s, 1100.9 tps, lat 21.266 ms stddev 6.349
progress: 2.0 s, 1253.0 tps, lat 18.779 ms stddev 7.913
progress: 3.0 s, 1219.0 tps, lat 20.083 ms stddev 13.212
```