RDS for PostgreSQL

Kernels

Issue	01
Date	2025-09-02

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

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

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

Contents

1 RDS for PostgreSQL Kernel Version Description

The following table lists the release dates and updates of community versions supported by RDS for PostgreSQL.

Table 1-1 Kernel version release history

Released In	Supported Community Versions	Updates
2025-05	17.5 16.9 15.13 14.18 13.21	Main updates: • Synchronized the latest code in the community to fix security vulnerabilities such as **CVE-2025-4207** and functional problems. More release notes: • **Release Notes on PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21**
2025-02	17.4 16.8 15.12 14.17 13.20	Main updates: • Synchronized the latest code in the community to fix security vulnerabilities such as **CVE-2025-1094** and functional problems. More release notes: • **Release Notes on PostgreSQL 17.4, 16.8, 15.12, 14.17, and 13.20**

Released In	Supported Community Versions	Updates
2024-11	16.6 15.10 14.15 13.18 12.22	Main updates: • Synchronized the latest code in the community to fix security vulnerabilities such as **CVE-2024-10978** and **CVE-2024-10979** and functional problems. • C functions are not allowed in RDS for PostgreSQL 16.6, 15.10, 14.15, 13.18, 12.22, and later. More release notes: • **Release Notes on PostgreSQL 16.6, 15.10, 14.15, 13.18, and 12.22** Security hardening: • Fixed the privilege escalation vulnerability caused by the third-party open-source extensions PL/Perl and PLV8.
2024-09	16.4 15.8 14.13 13.16 12.20	Main updates: • Synchronized the latest code in the community to fix security vulnerabilities such as **CVE-2024-7348** and **CVE-2024-4317** and functional problems. More release notes: • **Release Notes on PostgreSQL 16.4, 15.8, 14.13, 13.16, and 12.20**
2024-03	• 16.2 • 15.6 • 14.11 • 13.14 • 12.18	Main updates: • Introduced PostgreSQL 16. • Synchronized the latest code in the community to fix security vulnerabilities such as **CVE-2024-0985** and functional problems. More release notes: • **Release Notes on PostgreSQL 16** • **Release Notes on PostgreSQL 16.2, 15.6, 14.11, 13.14, and 12.18**

Released In	Supported Community Versions	Updates
2023-12	• 15.5 • 14.10 • 13.13 • 12.17 • 11.22	Main updates: • Synchronized the latest code in the community to fix security vulnerabilities such as **CVE-2023-5868**, **CVE-2023-5869**, and **CVE-2023-5870** and functional problems. More release notes: • **Release Notes on PostgreSQL 15.5, 14.10, 13.13, 12.17, and 11.22**
2023-09	• 15.4 • 14.9 • 13.12 • 12.16 • 11.21	Main updates: • Introduced PostgreSQL 15. • Synchronized the latest code in the community to fix security vulnerabilities such as **CVE-2023-34917** and **CVE-2023-39418** and functional problems. More release notes: • **Release Notes on PostgreSQL 15.4** • **Release Notes on PostgreSQL 14.9, 13.12, 12.16, and 11.21**
2023-08	• 14.8 • 13.11 • 12.15 • 11.20	Main updates: • Synchronized the latest code in the community to fix security vulnerabilities such as **CVE-2023-2454** and **CVE-2023-2455** and functional problems. More release notes: • **Release Notes on PostgreSQL 14.8, 13.11, 12.15, and 11.20** Security hardening: • Fixed the arbitrary code execution vulnerability caused by **lo_export()**.

Released In	Supported Community Versions	Updates
2023-04	• 14.6 • 13.9 • 12.13 • 11.18 • 10.23	Main updates: • Supported the **pgl_ddl_deploy** extension. • Upgraded the **zhparser** extension. More information about extensions: • Different kernel versions support different extensions. For details, see **Supported Extensions**. Security hardening: • Fixed the arbitrary code execution vulnerability caused by **copy from/to**.
2022-12	• 14.6 • 13.9 • 12.13 • 11.18 • 10.23	Main updates: • Synchronized the latest code in the community to fix security vulnerabilities and functional problems. More release notes: • **Release Notes on PostgreSQL 14.6, 13.9, 12.13, 11.18, and 10.23**
2022-07	• 14.4 • 13.7 • 12.11 • 11.16 • 10.21	Main updates: • Introduced PostgreSQL 14. • Synchronized the latest code in the community to fix security vulnerabilities such as **CVE-2022-1552** and functional problems. More release notes: • **Release Notes on PostgreSQL 14.4** • **Release Notes on PostgreSQL 13.7, 12.11, 11.16, and 10.21**
2022-04	• 13.6 • 12.10 • 11.15 • 10.20 • 9.6.24	Main updates: • Synchronized the latest code in the community to fix security vulnerabilities and functional problems. More release notes: • **Release Notes on PostgreSQL 13.6, 12.10, 11.15, 10.20, and 9.6.24**

Released In	Supported Community Versions	Updates
2021-04	• 13.2 • 12.6 • 11.11 • 10.16 • 9.6.21 • 9.5.25	Main updates: • Introduced PostgreSQL 13. • Synchronized the latest code in the community to fix security vulnerabilities and functional problems. • Supported failover slots. For details, see **Failover Slot for Logical Subscriptions**. More release notes: • **Release Notes on PostgreSQL 13.2, 12.6, 11.11, 10.16, 9.6.21, and 9.5.25**
2020-03	12.2	/
2019-12	9.5.19	/
2019-11	• 11.5 • 10.10 • 9.6.15	/
2019-08	• 11.4 • 9.6.13 • 9.5.17	/
2019-07	• 11.2 • 10.8	/
2019-06	9.5.15	/
2019-03	• 10.6 • 9.6.11	/
2019-01	11.0	/
2018-11	10.3	/
2018-03	9.6.5	/
2018-02	9.6.3	/
2017-10	9.5.5	/

2 Main Kernel Functions

2.1 Failover Slot for Logical Subscriptions

Scenarios

RDS for PostgreSQL synchronizes information about logical replication slots specified as failover slots from the primary DB instance to the standby DB instance to ensure logical subscriptions remain connected after a primary/standby switchover.

Information about logical replication slots will not be transferred to the new primary DB instance during a primary/standby switchover. If there is a primary/standby switchover, the logical subscriptions are disconnected, and you will need to manually create new slots. Failover slots make it possible to synchronize all logical slots from the primary DB instance to the standby DB instance, preventing logical subscriptions from being disconnected after a primary/standby switchover.

Precautions

- This function is supported in RDS for PostgreSQL 12.6 and later minor versions, and all minor versions of RDS for PostgreSQL 13 and later major versions.
- This function supports only logical subscriptions. It does not support physical subscriptions.
- This function introduces new log types. Therefore, if you enable this function for your instance, old slot information will be retained when you restore data to a new instance from backups. You need to manually delete the old slot information after the restoration is complete.

How to Use

Run the following SQL statement on the publishing client to create a failover slot:

Table 2-1 Commands and parameters

Version	Command	Parameters
RDS for PostgreSQL 12 and RDS for PostgreSQL 13	**SELECT * FROM pg_create_logical_replication_slot(***'slotname'*, *'pgoutput'*, **false, true);**	• *slotname* is the name of the logical slot. • *pgoutput* is the name of the plugin. You can also change it to another supported plugin. • The third parameter specifies if the slot is a temporary slot. • The fourth parameter specifies if the slot is a failover slot.
RDS for PostgreSQL 14 or later	**SELECT * FROM pg_create_logical_replication_slot(***'slotname'*, *'pgoutput'*, **false, false, true);**	• *slotname* is the name of the logical slot. • *pgoutput* is the name of the plugin. You can also change it to another supported plugin. • The third parameter specifies if the slot is a temporary slot. • The fourth parameter specifies whether to enable two-phase commit. • The fifth parameter specifies if the replication slot is a failover slot.

To create a failover slot, the value of the third parameter must be **false** and that of the last parameter must be **true**. If you do not specify the last parameter, the created slot is not a failover slot.

How to Check

Run the following SQL statement on the publishing client to query the replication slot information of the database:

select * from pg_get_replication_slots();

- Check the **failover** field. If the value is **true**, the replication slot is a failover slot.
- If there is no the **failover** field or its value is **false**, the replication slot is not a failover slot.

Example of a Complete Logical Subscription

- Create a table on the publishing client.

  create table tableName(id int primary key, num int);
- Create a publication on the publishing client.

  create publication pubName for table tableName;

- Create a failover slot on the publishing client. (For RDS for PostgreSQL 12 or 13, delete the fourth parameter.)

  SELECT * FROM pg_create_logical_replication_slot(*'slotname'***, *'pgoutput'*, false, false, true);**

- Insert data into the table on the publishing client.

  insert into tableName values(1,1);

  insert into tableName values(2,2);

- Create a table on the subscription client.

  create table tableName (id int primary key, num int);

- Create a subscription on the subscription client and specify the name of the failover slot.

  create subscription subName connection 'host=192.168.0.10 dbname=postgres user=root port=5432 password=*xxxxxxx***' publication pubName with(copy_data=true,create_slot=false,slot_name= slotname);**

- Query data on the subscription client and check whether the data is subscribed to.

  select * from tableName;

- Perform a primary/standby switchover.

  Insert data into the table on the publishing client and view the data on the subscription client. The logical subscription is not disconnected.

