TencentDB for PostgreSQL

User Manual

Product Documentation

Contents

User Manual
Limits
Instance Type

Last updated : 2020-02-26 12:27:48

Currently, the following instance types exist in Cloud Database PostgreSQL:

- **Master instance** : This instance can be purchased directly. It has all the features of a database.
- **Read-only instances [only support manual addition]**: A read-only instance cannot exist alone, and must belong to a master instance. A read-only instance only acquires data through synchronization with the master instance, and is invisible in the instance list.

Operational Constraints

Last updated : 2020-02-26 12:24:50

To provide secure and stable instance services, Cloud Database PostgreSQL stipulates the following usage constraints:

Operation	Permission Description
Database super admin permission	Super user permission is unavailable
Modify database parameter settings	Currently unavailable
Data recovery	Only support using psql to recover dump data created by pg_dump
Build database replication model	HA model is automatically built based on PostgreSQL stream replication, eliminating the need to build it manually. PostgreSQL Standby nodes are invisible to users and thus cannot be directly accessed

Monitoring Feature

Last updated : 2020-02-26 12:04:48

Console monitoring

To make it easy for users to view and grasp the running information of the instance, cloud database PostgreSQL provides a wealth of performance monitoring items, and users can log in PostgreSQL console To view it in the **system Monitoring** tab of the corresponding instance management page.

Monitoring Metrics

Metric's Chinese name	Metric's English name	Metric's meaning	Metric unit
CPU Utilization	Cpu	Instance CPU utilization. Due to the flexible CPU restriction policy adopted in idle time, the CPU utilization may be greater than 100%.	%
Used Storage	Storage	The instance occupies the available space of the disk	GB
Disk IOPS	Iops	IOPS of the instance (number of requests per second)	times/sec
Inbound Traffic	In_flow	Traffic inputted by reading and writing of an example	KB/ seconds
Outbound Traffic	Out_flow	Traffic of example read-write output	KB/ seconds
Connections	Connections	Historical trend of active connections of instances	__item(s)
Requests	Read_write_calls	Total number of read and write (CRUD) requests per minute	/min
Number of read requests	Read_calls	Total number of read requests per minute	/min
Write Requests	Write_calls	Total number of write requests per minute	/min

Metric's Chinese name	Metric's English name	Metric's meaning	Metric unit
Number of other requests	Other_calls	Total number of requests except read and write (for example, Drop), accumulates by minute	/min
Buffer Cache Hit Rate	Hit_percent	Data cache hit ratio	%
Average Execution Latency	Sql_runtime_avg	Average execution time of all SQL requests, excluding SQL in the transaction	ms
Longest TOP10 executive Latency	Sql_runtime_avg	Average SQL of the TOP10 with the longest execution time	ms
The shortest TOP10 executive Latency	Sql_runtime_min	The average SQL of the TOP10 with the shortest execution time	ms
Number of remaining XID	Remain_xid	Number of remaining Transaction IDs. There are a maximum of 2 ^ 32 Transaction Ids. It is recommended to execute "vacuum full" manually if the number falls below 1000000	__item(s)
Synchronization difference between master/slave XLOG	Xlog_diff	(Sample is taken every minute) The synchronization difference between the master XLOG and the slave XLOG. This represents synchronization delay, and lower is better	byte

Pg_stat_statements View

You can also use the Pg_stat_statements View query pg detailed performance Metric. The pg_stat_statements module provides a method to track the execution statistics of all SQL statements executed by a server, which can be used to count the resource cost of the database and analyze TOP SQL.

```
select * from pg_stat_statements;
```

Alarming

Last updated : 2020-02-26 11:52:08

Operation Scenarios

The alarming feature is used to trigger alarms and send alarm messages when the status of a TencentDB instance changes. The created alarm can periodically determine whether an alarm notification should be sent based on the difference between the monitored metric and the given threshold.

You can take appropriate precautionary or remedial measures timely when an alarm is triggered. Therefore, properly created alarms can help you improve the robustness and reliability of your applications. For more information, please see Alarm Configuration in Cloud Monitor.

If you want to send an alarm message for a specific status of a product, you need to create an alarm policy first, which is composed of three mandatory components: name, type, and alarm trigger. Each alarm policy is a set of alarm triggers in the logical OR relationship, i.e., as long as one of the triggers is satisfied, the alarm will be triggered. The alarm notification will be sent to all users associated with the alarm policy. They can take appropriate actions after receiving the notification.

> Please make sure that you have set the default alarm recipient; otherwise, the default alarm policy of TencentDB won't be able to send notifications.

Directions

Step 1. Create an alarm policy

1. Log in to the Cloud Monitor Console and select **Alarm Configuration** > **Alarm Policy** on the left sidebar.
2. In the alarm policy list, click **Create**.
3. Set the policy name, policy type, target product, alarm object, and trigger.
   - An alarm trigger is a semantic condition composed of metric, comparison, threshold, statistical period, and duration. For example, if trigger conditions are configured as follows: the metric is disk utilization, the comparison is >, the threshold is 80%, the statistical period is 5 minutes, and the duration is two statistical periods, then the data on disk utilization of a database will be

collected by Cloud Monitor once every five minutes, and an alarm will be triggered if the disk utilization exceeds 80% for two consecutive times.
- The object instance to be associated with can be found by selecting the region where the object is located or searching for the instance ID of the object.

4. After confirming that everything is correct, click **Complete**.

Step 2. Associate with an object

After the alarm policy is created, you can associate alarm objects with it. When an alarm object satisfies an alarm trigger, an alarm notification will be sent.

1. In the alarm policy list, click the name of an alarm policy to enter the alarm policy management page.
2. On the alarm policy management page, click **Create Object**.
3. Select a desired Tencent Cloud product and click **Apply** to associate it with the alarm policy.

Step 3. Set an alarm recipient

Alarm recipients are those who will receive alarm messages.

1. In the alarm policy list, click the name of an alarm policy.
2. On the alarm policy management page, select **Alarm Recipient** and click **Edit**.
3. Select the user group to be notified, set relevant options, and click **Save**.

Slow Log Analysis

Last updated : 2020-02-21 15:17:41

Feature Description

An SQL statement query that takes more time than the specified value is referred to as a "slow log", and the corresponding statement is called a "slow log statement". The process where a database administrator (DBA) analyzes slow log statements and finds out the reasons why slow logs occur is known as "slow log analysis".

You can log in to the TencentDB for PostgreSQL Console and go to the **Performance Optimization** module on the instance management page to perform slow query analysis.

Descriptions of Main Parameters

Default settings of slow log analysis

- **Slow log feature**: enabled by default.
- **Slow SQL log time (log_min_duration_statement)**: 1 second by default, that is, only slow log statements that exceed 1 second will be logged. If you need to adjust the time, please submit a ticket with the instance ID and the desired time value.
- **Analyzed data output latency**: 1–5 minutes.
- **Log retention period**: 7 days (up to the last 10,000 records)

Descriptions of analysis list fields

Currently, the slow log analysis feature provides the following information of slow SQL statements:

Basic information

- **Last execution time**: the time when the abstract statement appeared for the last time within the specified time range. As some statements may be executed for a long time, all statements are logged by the `begin_time` of statement execution.
- **Abstracted SQL statement**: the statement after constants are removed from the slow SQL. The abstracted statement can be used for aggregated statistics of similar statements to facilitate your analysis.
- **Database**: the database called by the statement.
- **Account**: the account used by the statement.

- **First execution time**: the time when the slow SQL statement appeared for the first time within the specified time range (there may be many records after abstraction and aggregation).
- **Total occurrences**: how many times in total the slow SQL statement appeared within the specified time range.
- **Occurrence percentage**: occurrence percentage of a slow log statement out of all slow log statements within the specified time range.
- **Number of rows sent**: total number of data rows sent (returned) to the client by the database server within the specified time range.

Query statistics

- **Total time**: total time of a slow log statement within the specified time range.
- **Total time percentage:** time percentage of a slow log statement out of all slow log statements within the specified time range.
- **Average time**: average time calculated by dividing the total time of a slow log statement by total number of occurrences.
- **Minimum time**: minimum occurrence time of a slow SQL in the abstracted statement, which is used to determine whether the statement is sporadic.
- **Maximum time**: maximum occurrence time of a slow SQL in the abstracted statement, which is used to determine whether the statement is sporadic.

Data read/write statistics

- **Blocks of shared memory read**: how much data in the shared memory is read by the slow SQL statement within the specified time range.
- **Blocks of shared memory written**: how much data is written to the shared memory by the slow SQL within the specified time range.

IO read/write time statistics

- **Total IO read time**: total time of IO reads by the slow SQL statement within the specified time range, which is used to help determine whether the statement performed time-consuming operations such as full scan.
- **Total IO write time**: total time of IO writes by the slow SQL within the specified time range, which is used to help determine whether the statement writes large amounts of (temporary) data at a time.

Error Logs

Last updated : 2020-02-21 15:18:08

1. Feature Description

Error logs refer to logs generated due to operation, SQL, and system errors during database running. Error logs are usually used to find out the causes of errors in business systems or databases.

You can go to the TencentDB for PostgreSQL Console > **Instance Management** > **Performance Optimization** to view error logs.

2. Default Settings of Error Log

Error log feature: enabled by default

Error log level: log_min_error_statement=ERROR.

Analyzed data output latency: 1–5 minutes.

Log retention period: 7 days (up to the last 10,000 records).

Backing up Data

Last updated : 2020-02-21 15:18:35

Operation Scenarios

Currently, TencentDB for PostgreSQL High Availability Edition only supports physical backup:

- **Full backup**: once a day at 01:00.
- **Incremental backup**: for xlog files, backup will be performed once every 15 minutes or when the number of files reaches 60.
- **Data file retention period**: full and incremental backups in the last 7 days will be retained.

This document describes how to download backup files in the TencentDB for PostgreSQL Console.

Directions

1. Log in to the TencentDB for PostgreSQL Console and click an instance name to enter the instance details page.
2. On the **Backup Management** tab, click **Backup List** or **xlog List**, select the desired backup, and click **Download** in the "Operation" column.

   > Backup data in the **Backup List** is full backup, while that in the **xlog List** is incremental backup.

3. In the pop-up window, you can choose a VPC address or public network address to download the backup file.

   > To ensure data security, an address will be valid for 15 minutes. After it expires, please refresh the page to get a new address. Please access a VPC address in a VPC.

Restoring PostgreSQL Data on CVM

Last updated : 2020-02-21 15:25:58

Downloading Backup in the Console for Restoration

1. Install PostgreSQL

In the CVM instance where data is to be restored, install PostgreSQL on the same version as that of the backup data. If PostgreSQL has already been installed, this step can be skipped.

> This document shows you how to install PostgreSQL 10 and restore data on a CentOS 7-based CVM instance.

1. Log in to a Linux-based CVM instance. For more information, please see Getting Started with Linux CVM.
2. Install PostgreSQL. The yum repository method is used in this document. You can click here to find the needed yum repository.
   Run the following command to install PostgreSQL 10:

```
yum install https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm
yum install postgresql10-server postgresql10-contrib postgresql10 postgresql10.x86_64
```

> The command for installing PostgreSQL 9.5 is as follows:

```
yum install https://yum.postgresql.org/9.5/redhat/rhel-7.6-x86_64/pgdg-centos95-9.5-3.noarch.rpm
yum install postgresql95-server postgresql95-contrib postgresql95
```

3. Run the following command to check the installation result:

```
rpm -aq| grep postgres
```

   A message similar to the one below will be returned:

```
[root@i-87-575-VM vmuser]# rpm -aq| grep postgres
postgresql10-libs-10.11-2PGDG.rhel7.x86_64
```

```
postgresql10-server-10.11-2PGDG.rhel7.x86_64
postgresql10-contrib-10.11-2PGDG.rhel7.x86_64
postgresql10-10.11-2PGDG.rhel7.x86_64
```

2. Create a recovery directory as postgres user

Switch to the postgres user and create a recovery directory in the CVM instance

```
mkdir /var/lib/pgsql/10/recovery
```

`recovery` is an example directory name, which can be modified as needed. In the following examples, the directory names will be the same for one major version. For example, the directory will be `/var/lib/pgsql/10` for PostgreSQL 10.x and `/var/lib/pgsql/9.5` for PostgreSQL 9.5.x.

> The command for PostgreSQL 9.5 is as follows:

```
mkdir /var/lib/pgsql/9.5/recovery
```

3. Download the full backup file

1. Log in to the TencentDB for PostgreSQL Console, select the desired instance in the instance list, and click **Manage** in the "Operation" column to enter the management page.
2. On the **Backup Management** tab, select the backup version to be restored based on backup time and click **Download** in the "Operation" column.
3. Download the backup file from the provided VPC address or public network address.

> - If a VPC address is to be used, the TencentDB instance and CVM instance should be in the same VPC, and the backup needs to be downloaded to the `/var/lib/pgsql/10/recovery` directory.
> - If a public network address is to be used, the downloaded backup file needs to be uploaded to the `/var/lib/pgsql/10/recovery` directory in the CVM instance.

After upload, the following information will be displayed:

```
[root@VM_0_12_centos recovery]# ls -lh
total 6.3M
-rw-r--r-- 1 root root 6.3M Dec 23 11:45 20191221010146.tar.gz
```

4. Decompress the full backup file

Run the following command to decompress the full backup file:

```
cd /var/lib/pgsql/10/recovery
tar -xf 20191221010146.tar.gz
```

After decompression, the following information will be displayed:

```
[root@VM_0_12_centos recovery]# ls -lh
total 6.4M
-rw-r--r-- 1 root root  6.3M Dec 23 11:45 20191221010146.tar.gz
-rw------- 1 1003 users  215 Dec 21 01:01 backup_label
drwx------ 7 1003 users 4.0K Dec 13 17:37 base
-rw------- 1 1003 users   35 Dec 21 00:00 current_logfiles
drwx------ 2 1003 users 4.0K Dec  5 20:12 global
drwx------ 2 1003 users 4.0K Dec  2 21:59 pg_commit_ts
drwx------ 2 1003 users 4.0K Dec  2 21:59 pg_dynshmem
-rw------- 1 1003 users 4.8K Dec  2 21:59 pg_hba.conf
-rw------- 1 1003 users 1.6K Dec  2 21:59 pg_ident.conf
drwx------ 4 1003 users 4.0K Dec 21 01:01 pg_logical
drwx------ 4 1003 users 4.0K Dec  2 21:59 pg_multixact
drwx------ 2 1003 users 4.0K Dec  2 21:59 pg_notify
drwx------ 2 1003 users 4.0K Dec  2 21:59 pg_replslot
drwx------ 2 1003 users 4.0K Dec  2 21:59 pg_serial
drwx------ 2 1003 users 4.0K Dec  2 21:59 pg_snapshots
drwx------ 2 1003 users 4.0K Dec  2 21:59 pg_stat
drwx------ 2 1003 users 4.0K Dec 21 01:01 pg_stat_tmp
drwx------ 2 1003 users 4.0K Dec  2 21:59 pg_subtrans
drwx------ 2 1003 users 4.0K Dec  2 21:59 pg_tblspc
drwx------ 2 1003 users 4.0K Dec  2 21:59 pg_twophase
-rw------- 1 1003 users    3 Dec  2 21:59 PG_VERSION
drwx------ 2 1003 users 4.0K Dec  2 21:59 pg_xact
drwxr-xr-x 2 1003 users 4.0K Dec  3 01:01 pg_xlog
-rw------- 1 1003 users   11 Dec 20 12:02 pg_xlog_archive.tmp
-rw------- 1 1003 users   88 Dec  2 21:59 postgresql.auto.conf
-rw------- 1 1003 users  24K Dec  2 21:59 postgresql.conf
```

5. Remove unnecessary temporary files

Run the following command to remove unnecessary temporary files:

```
rm -rf backup_label
```

6. Modify the configuration file

1. Comment out the following options in the `postgresql.conf` configuration file by using # at the beginning of the line.
   Comment all out if there are multiple such options.

   ```
   shared_preload_libraries
   local_preload_libraries
   pg_stat_statements.max
   pg_stat_statements.track
   ```

```
archive_mode
archive_command
synchronous_commit
synchronous_standby_names
```

2. Modify the `postgresql.conf` configuration file.

```
port = '5432' ## Change the value of the `port` parameter to 5432
unix_socket_directories = '/var/run/postgresql/' ## Change the value of `unix_socket_directories` to `/var/run/postgresql/`; this step can be skipped if the value is not set
```

3. Append configurations to the `postgresql.conf` configuration file, indicating that the strong sync mode will no longer be used.

```
synchronous_commit = local
synchronous_standby_names = ''
```

7. Modify folder permissions as root user

```
chmod 0700 /var/lib/pgsql/10/recovery
chown postgres:postgres /var/lib/pgsql/10/recovery -R
```

After modification, the following information will be displayed:

```
[root@VM_0_12_centos recovery]# ls -al
total 6528
drwx------ 19 postgres postgres    4096 Dec 23 11:50 .
drwx------  6 postgres postgres    4096 Dec 23 11:44 ..
-rw-r--r--  1 postgres postgres 6546935 Dec 23 11:45 20191221010146.tar.gz
-rw-------  1 postgres postgres     215 Dec 21 01:01 backup_label
drwx------  7 postgres postgres    4096 Dec 13 17:37 base
-rw-------  1 postgres postgres      35 Dec 21 00:00 current_logfiles
drwx------  2 postgres postgres    4096 Dec  5 20:12 global
drwx------  2 postgres postgres    4096 Dec  2 21:59 pg_commit_ts
drwx------  2 postgres postgres    4096 Dec  2 21:59 pg_dynshmem
-rw-------  1 postgres postgres    4858 Dec  2 21:59 pg_hba.conf
```

8. Use the incremental backup file (optional)

If this step is skipped, the content of the database is that when the full backup was started.
Put the xlog files in the `/var/lib/pgsql/10/recovery/pg_wal` folder; if the downloaded backup does not contain the `pg_wal` directory, please modify `pg_xlog` to `pg_wal`, and PostgreSQL will automatically replay the xlog files.
For example, if a full backup is started at 12:00 and all xlog files between 12:00 and 13:00 are put in the `pg_wal` folder, then data can be restored to 13:00.

For PostgreSQL 9.x, the folder is `/var/lib/pgsql/9.x/recovery/pg_xlog`.

1. On the **Backup Management** page in the console, get the xlog download address and download the incremental backup file (xlog).
   After download, the following information will be displayed:

```
[root@VM_0_12_centos recovery]# mv pg_xlog pg_wal
[root@VM_0_12_centos recovery]# cd pg_wal/
[root@VM_0_12_centos pg_wal]# ls -lh
total 64K
-rw-r--r-- 1 root root 31K Dec 23 11:44 20191221010157_20191221010157.tar.gz
-rw-r--r-- 1 root root 31K Dec 23 11:44 20191221013157_20191221013157.tar.gz
```

2. Decompress the log to the `pg_wal` folder.

```
tar -xf 20170904010214_20170905010205.tar.gz
```

```
[root@VM_0_12_centos pg_wal]# ll
total 32836
-rw------- 1 postgres postgres 16777216 Dec 21 01:01 000000010000000000000002D
-rw------- 1 postgres postgres      312 Dec 21 01:01 000000010000000000000002D.00000028.backup
-rw------- 1 postgres postgres 16777216 Dec 21 01:31 000000010000000000000002E
-rw-r--r-- 1 postgres postgres    31319 Dec 23 11:44 20191221010157_20191221010157.tar.gz
-rw-r--r-- 1 postgres postgres    30966 Dec 23 11:44 20191221013157_20191221013157.tar.gz
```

9. Start PostgreSQL as postgres user

```
/usr/pgsql-10/bin/pg_ctl start -D /var/lib/pgsql/10/recovery
```

```
-bash-4.2$ /usr/pgsql-10/bin/pg_ctl start -D /var/lib/pgsql/10/recovery
waiting for server to start....2019-12-23 11:59:42.654 CST [14061] LOG:  listening on IPv4 address "0.0
.0.0", port 5432
2019-12-23 11:59:42.654 CST [14061] LOG:  listening on IPv6 address "::", port 5432
2019-12-23 11:59:42.664 CST [14061] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2019-12-23 11:59:42.686 CST [14061] LOG:  listening on Unix socket "/tmp/.s.PGSQL.5432"

2019-12-23 11:59:42.840 CST [14061] LOG:  redirecting log output to logging collector process
2019-12-23 11:59:42.840 CST [14061] HINT:  Future log output will appear in directory "pg_log".
 done
server started
```

10. Log in to PostgreSQL

1. Log in to PostgreSQL.

```
export PGDATA=/var/lib/pgsql/10/recovery
psql
```

```
-bash-4.2$ export PGDATA=/var/lib/pgsql/10/recovery
-bash-4.2$ psql
psql (9.2.24, server 10.11)
WARNING: psql version 9.2, server version 10.0.
         Some psql features might not work.
Type "help" for help.
```

2. Check whether the database is running.

```
/usr/pgsql-10/bin/pg_ctl status -D /var/lib/pgsql/10/recovery
```

If the prompt is "server is running", the database is running.

```
-bash-4.2$ /usr/pgsql-10/bin/pg_ctl status -D /var/lib/pgsql/10/recovery
pg_ctl: server is running (PID: 14061)
/usr/pgsql-10/bin/postgres "-D" "/var/lib/pgsql/10/recovery"
```

Manually Exporting Data for Restoration

You can also manually export backup data and then restore it on CVM. This scheme is applicable to both Windows and Linux regardless of the file system where physical files reside.

1. Dump the data on CVM as shown below:

```
Command format: pg_dump -h <access IP> -U <accessing user> -f <full path to the backup file> -c -C <name of the exported database>
Example:
/usr/pgsql-10/bin/pg_dump -h 192.168.0.16 -U testroot -f backup.sql -c -C postgres
```

- If no file format is specified, a text file will be exported by default, as shown below:
  ```

# -- PostgreSQL database dump

-- Dumped from database version 9.5.4
-- Dumped by pg_dump version 9.5.19
SET statement_timeout = 0;
SET lock_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
  ```

SET check_function_bodies = false;

SET xmloption = content;

SET client_min_messages = warning;

SET row_security = off;

```
- If there is a massive amount of data, specify the file format as binary file by using `-Fc`.
2. Restore the data on CVM.
- For text files, data can be restored by running the following SQL statement:
```

psql -U postgres <./backup.sql

```
>As there are extension like pg_stat_error, an error may occur, but that does not affect data import.
- For binary files, data needs to be restored by using `pg_restore`.
```

Migrating Data

Last updated : 2020-02-21 15:23:09

Using Tencent Cloud DTS

The **data migration** feature of Tencent Cloud **Data Transmission Service (DTS)** can be used to import data to TencentDB for PostgreSQL.

For more information, please see Data Migration.

Using `psql` Command

If your data is already on PostgreSQL, you can easily migrate it to TencentDB for PostgreSQL by using the `psql` command.
Data migration mainly involves two parts:

1. Perform logical backup by using the `pg_dump` command to create dump data.
2. Restore the backup data to TencentDB for PostgreSQL.
   Preparation: get the PostgreSQL instance ready. If not, please see the operation guide on applying for an instance and connecting to it.

Step 1

Connect to the local database by using the PostgreSQL client and run the following command to back up the data.

```
pg_dump -U username -h hostname -p port -d databasename -f filename
```

Descriptions of parameters:

Parameter	Description
username	Username of the local database
hostname	Host name of the local database. You can use `localhost` if you log in from a local database host
port	Port number of the local database

Parameter	Description
databasename	Name of the local database to be backed up
filename	Name of the backup file to be generated, such as `mydump.sql`

Step 2

Preparation: upload the backup data to the CVM instance in the same private network and then restore the data over the private network, which helps ensure network stability and data security. After you log in to the CVM instance, run the following command on the PostgreSQL client to restore the data.

```
psql -U username -h hostname -d desintationdb -p port -f dumpfilename
```

Descriptions of parameters:

Parameter	Description
username	Username of a TencentDB for PostgreSQL database
hostname	Address of a TencentDB for PostgreSQL database
desintationdb	Name of a TencentDB for PostgreSQL database
port	Port number of a TencentDB for PostgreSQL database
dumpfilename	Filename of the backup data, such as `mydump.sql`

Restarting Instances

Last updated : 2020-02-26 12:32:22

Restart is indispensable to the maintenance of databases. Restarting a PostgreSQL instance is equivalent to restarting a database (service and process) on a local server.

Restarting an Instance on the Console

1) Log in to CDB for PostgreSQL Console.

2) Click **Restart** in the operation column on the instance list to restart a single running PostgreSQL instance.

Notes about Database Restart

1) Please exercise great caution when restarting a database, which plays a vital role in the business. Before the restart, it is recommended to disconnect the database from server and **stop writing** data.

2) Generally, it takes a dozen seconds to a few minutes to wait until the restart is completed. The instance cannot provide any service during the process.

3) **There is a possibility of failure of database restart**, which is **normal**. In case of a failure of restart, you can (manually) click **Restart** to try again.

4) Staring an instance does not change any of its physical attributes, so the public IP, private IP, and any data stored on the instance will remain unchanged.

5) After the restart, reconnection to the database is needed. Please make sure your business has a reconnection mechanism.

Supported Extensions

Last updated : 2020-02-26 12:30:39

TencentDB for PostgreSQL supports creating custom plugins (create extension pluginName). The following plugins are supported (some plugins are no longer supported in PostgreSQL 10 due to its more powerful features):

Extension	9.3.5	9.5.4	10.4
btree_gin	✓	✓	✓
btree_gist	✓	✓	✓
chkpass	✓	✓	✓
citext	✓	✓	✓
cube	✓	✓	✓
dict_int	✓	✓	✓
earthdistance	✓	✓	✓
postgres_fdw	✓	✓	×
postgres_fdw	✓	✓	×
postgres_fdw	✓	✓	×
fuzzystrmatch	✓	✓	✓
hstore	✓	✓	✓
intagg	✓	✓	✓
intarray	✓	✓	✓
isn	✓	✓	✓
jsonbx	×	✓	×
ltree	✓	✓	✓
pg_hint_plan	✓	✓	×
pg_prewarm	×	✓	✓

Extension	9.3.5	9.5.4	10.4
pg_stat_statements	✓	✓	✓
pg_trgm	✓	✓	✓
pgcrypto	✓	✓	✓
pgrouting	✓	✓	✓
pgrowlocks	✓	✓	✓
pl/perl	✓	✓	✓
pl/pgsql	✓	✓	✓
pl/tcl	✓	✓	✓
plcoffee	✓	✓	✓
plls	✓	✓	✓
plv8	✓	✓	✓
postgis	✓	✓	✓
postgis_tiger_geocoder	✓	✓	✓
postgres_fdw	✓	✓	✓
sslinfo	✓	✓	✓
tablefunc	✓	✓	✓
tsearch2	✓	✓	×
unaccent	✓	✓	✓
uuid-ossp	✓	✓	✓
zhparser	✓	✓	✓

Foreign Data Wrapper

Last updated : 2020-02-21 15:44:20

Overview

Foreign Data Wrapper (FDW) is a type of extensions provided by PostgreSQL for accessing external data sources, including other databases in the current instance or other instances.
Based on the type of the target database instance, FDW extensions come in different forms, such as postgres_fdw, mysql_fdw, and mongo_fdw. The steps to use FDW are as follows:

1. Run the `CREATE EXTENSION` statement to install the FDW extension.
2. Run the `CREATE SERVER` statement to create a foreign server object for each remote database to be connected to, and specify connection information except `user` and `password` as the options for the server.
3. Run the `CREATE USER MAPPING` statement to create a user mapping for each database to be accessed through the foreign server, and specify the remote database's username and password as the `user` and `password` of the mapped user.
4. Run the `CREATE FOREIGN TABLE` statement to create a foreign table for each remote table to be accessed. The corresponding columns of the foreign table must match those of the remote table. You can also use different table and column names in the foreign table than the remote table, provided that you have the correct remote object name as an option to create the foreign table object.

As FDW supports cross-instance access, for security reasons, TencentDB for PostgreSQL optimizes access control over the creation of foreign server objects and implements categorized management based on the environment of the target instance. Auxiliary parameters are added to the open-source edition to verify user identity and adjust network policies.

Auxiliary Parameters for CREATE SERVER

1. Auxiliary parameters for extensions such as postgres_fdw and mysql_fdw

- host
  IP of the target instance, which is used for postgres_fdw. This parameter is required.
- address
  IP of the target instance, which is used for mysql_fdw. This parameter is required.

- port
  Port of the target instance. This parameter is required.
- instanceid
  Resource ID of the target instance. This parameter is required.
  i. If the target instance type is TencentDB, the parameter value will be the instance ID in the format of `postgres-xxxxx` or `mysql-xxxxx`, which can be viewed in the console.
  ii. If the target instance is on CVM, the parameter value will be the CVM instance ID in the format of `ins-xxxxx`.
- access_type
  Type of the target instance. This parameter is optional:
  1: the target instance is a TencentDB for PostgreSQL or TencentDB for MySQL instance; if no other types are explicitly specified, this will be the default type.
  2: the target instance is on CVM.
  3: the target instance is a self-built instance with public IP in Tencent Cloud.
  4: the target instance is connected via Tencent Cloud VPN.
  5: the target instance is connected via a self-built VPN.
  6: the target instance is connected via Direct Connect.
  7: the target instance is COS data.
- uin
  ID of the account to which the instance belongs, which is needed for verifying user permissions. This parameter is optional. For more information, please see Querying uin.
- own_uin
  ID of the root account to which the instance belongs, which is also needed for verifying user permissions. This parameter is optional.
- vpcid
  VPC ID. This parameter is optional; however, if the target instance is in a VPC of CVM, it will be required and can be viewed in the VPC Console.
- subnetid
  VPC subnet ID. This parameter is optional; however, if the target instance is in a VPC of CVM, it will be required and can be viewed on the subnet page in the VPC Console.
- dcgid
  Direct connect ID. This parameter is optional; however, it will be required if the target instance needs to be connected through Direct Connect.
- vpngwid
  VPN gateway ID. This parameter is optional; however, it will be required if the target instance needs to be connected through a VPN.
- region
  Region where the target instance resides; for example, "ap-guangzhou" represents the Guangzhou

region. This parameter is optional; however, it is required for cross-region access.

2. Auxiliary parameters for COS_FDW

- host
  Access domain name of COS, such as: `https://xxxx-xxxxxx.cos.ap-beijing.myqcloud.com`. This parameter is required.
- bucket
  COS bucket name. This parameter is required.
- id
  `SecretId` value of a TencentCloud API key, which is required and can be queried on the TencentCloud API key page in the CAM Console.
- key
  `SecretKey` value of a TencentCloud API key, which is required and can be queried on the TencentCloud API key page in the CAM Console.

3. Support for FDW

Name	Available Directly	Cross-region Use
postgres_fdw	Instances created before April 26, 2018 can use it after being restarted, while new instances can use it directly	Not supported by default. You can submit a ticket for application
mysql_fdw	Instances created before April 26, 2018 can use it after being restarted, while new instances can use it directly	Not supported by default. You can submit a ticket for application
cos_fdw	It is in beta test. You can submit a ticket for application	Not supported by default. You can submit a ticket for application

4. References

CREATE SERVER (9.3)
CREATE SERVER (9.5)

Example of Using postgres_fdw

postgres_fdw can be used to access data from other databases in the current instance or other instances.

1. Prerequisites

1. Create test data in the current instance.

```
postgres=>create role user1 with LOGIN CREATEDB PASSWORD 'password1';
postgres=>create database testdb1;
CREATE DATABASE
```

2. Create test data in the target instance.

```
postgres=>create role user2 with LOGIN CREATEDB PASSWORD 'password2';
postgres=> create database testdb2;
CREATE DATABASE
postgres=> ¥c testdb2 user2
You are now connected to database "testdb2" as user "user2".
testdb2=> create table test_table2(id integer);
CREATE TABLE
testdb2=> insert into test_table2 values (1);
INSERT 0 1
```

2. Create postgres_fdw

```
# Create
postgres=> ¥c testdb1
You are now connected to database "testdb1" as user "user1".
testdb1=> create extension postgres_fdw;
CREATE EXTENSION
# View
testdb1=> ¥dx
List of installed extensions
Name | Version | Schema | Description
-------------+---------+------------+------------------------------------------------------------
plpgsql | 1.0 | pg_catalog | PL/pgSQL procedural language
postgres_fdw | 1.0 | public | foreign-data wrapper for remote PostgreSQL servers
(2 rows)
```

3. Create a server

1. The target instance is a TencentDB instance.

```
# Access the data of `testdb2` in the target instance from `testdb1` in the current instance
testdb1=>create server srv_test1 foreign data wrapper postgres_fdw options (host 'xxx.xxx.xxx.
xxx',dbname 'testdb2', port '5432', instanceid 'postgres-xxxxx');
CREATE SERVER
```

2. The target instance is on CVM, and the network is the basic network.

```
testdb1=>create server srv_test foreign data wrapper postgres_fdw options (host 'xxx.xxx.xxx.xxx', dbname 'testdb2', port '5432', instanceid 'ins-xxxxx', access_type '2', region 'ap-guangzhou', uin 'xxxxxx', own_uin 'xxxxxx');
CREATE SERVER
```

3. The target instance is on CVM, and the network is VPC.

```
testdb1=>create server srv_test1 foreign data wrapper postgres_fdw options (host 'xxx.xxx.xxx.xxx',dbname 'testdb2', port '5432', instanceid 'ins-xxxxx', access_type '2', region 'ap-guangzhou', uin 'xxxxxx', own_uin 'xxxxxx', vpcid 'vpc-xxxxxx', subnetid 'subnet-xxxxx');
CREATE SERVER
```

4. The target instance is a self-built instance with public IP in Tencent Cloud.

```
testdb1=>create server srv_test1 foreign data wrapper postgres_fdw options (host 'xxx.xxx.xxx.xxx',dbname 'testdb2', port '5432', access_type '3', region 'ap-guangzhou', uin 'xxxxxx', own_uin 'xxxxxx');
CREATE SERVER
```

5. The target instance is connected via Tencent Cloud VPN.

```
testdb1=>create server srv_test1 foreign data wrapper postgres_fdw options (host 'xxx.xxx.xxx.xxx',dbname 'testdb2', port '5432', access_type '4', region 'ap-guangzhou', uin 'xxxxxx', own_uin 'xxxxxx', vpngwid 'xxxxxx');
```

6. The target instance is connected via a self-built VPN.

```
testdb1=>create server srv_test1 foreign data wrapper postgres_fdw options (host 'xxx.xxx.xxx.xxx',dbname 'testdb2', port '5432', access_type '5', region 'ap-guangzhou', uin 'xxxxxx', own_uin 'xxxxxx', vpngwid 'xxxxxx');
```

7. The target instance is connected via Direct Connect.

```
testdb1=>create server srv_test1 foreign data wrapper postgres_fdw options (host 'xxx.xxx.xxx.xxx',dbname 'testdb2', port '5432', access_type '6', region 'ap-guangzhou', uin 'xxxxxx', own_uin 'xxxxxx', dcgid 'xxxxxx');
CREATE SERVER
```

4. Create a user mapping

```
testdb1=> create user mapping for user1 server srv_test1 options (user 'user2',password 'password
2');
CREATE USER MAPPING
```

5. Create a foreign table

```
testdb1=> create foreign table foreign_table1(id integer) server srv_test1 options(table_name 'te
st_table2');
CREATE FOREIGN TABLE
```

6. Access foreign data

```
testdb1=> select * from foreign_table1;
id
----
1
(1 row)
```

7. Notes on using postgres_fdw

If the target instance is on CVM, please note the following:

1. The hba in PostgreSQL needs to be modified to allow the created mapped user (e.g., user2) to access via MD5. For more information on how to modify hba, please see PostgreSQL's official documentation.
2. If the target instance is not a TencentDB instance and the hot backup mode is configured for it, you need to update the server connection address or create another server after a master/slave switchover.

8. References

postgres_fdw Overview
CREATE SERVER (9.3)
CREATE SERVER (9.5)
pg_hba Overview (9.3)
pg_hba Overview (9.5)

Example of Using mysql_fdw

mysql_fdw can be used to access data from other MySQL instances.

1. Prerequisites

1. Create test data in the current instance.

```
postgres=>create role user1 with LOGIN CREATEDB PASSWORD 'password1';
postgres=>create database testdb1;
CREATE DATABASE
```

2. Create test data in the target instance.

```
postgres=>create role user2 with LOGIN CREATEDB PASSWORD 'password2';
postgres=> create database testdb2;
CREATE DATABASE
postgres=> ¥c testdb2 user2
You are now connected to database "testdb2" as user "user2".
testdb2=> create table test_table2(id integer);
CREATE TABLE
testdb2=> insert into test_table2 values (1);
INSERT 0 1
```

2. Create mysql_fdw

```
# Create
postgres=> ¥c testdb1
You are now connected to database "testdb1" as user "user1".
testdb1=> create extension mysql_fdw;
CREATE EXTENSION
# View
testdb1=> ¥dx
List of installed extensions
Name | Version | Schema | Description
-------------+---------+------------+---------------------------------------------------------
plpgsql | 1.0 | pg_catalog | PL/pgSQL procedural language
mysql_fdw | 1.0 | public | Foreign data wrapper for querying a MySQL server
(2 rows)
```

3. Create a server

1. The target instance is a TencentDB instance.

```
# Access the data of `testdb2` in the target instance from `testdb1` in the current instance
testdb1=>create server srv_test1 foreign data wrapper mysql_fdw options (host 'xxx.xxx.xxx.xx
x', port '3306', instanceid 'cdb-xxxxx', uin 'xxxxxx', region 'ap-guangzhou');
CREATE SERVER
```

2. The target instance is on CVM, and the network is the basic network.

```
testdb1=>create server srv_test foreign data wrapper mysql_fdw options (host 'xxx.xxx.xxx.xxx'
, port '3306', instanceid 'ins-xxxxx', access_type '2', region 'ap-guangzhou', uin 'xxxxxx', o
wn_uin 'xxxxxx');
CREATE SERVER
```

3. The target instance is on CVM, and the network is VPC.

```
testdb1=>create server srv_test1 foreign data wrapper mysql_fdw options (host 'xxx.xxx.xxx.xx
x', port '3306', instanceid 'ins-xxxxx', access_type '2', region 'ap-guangzhou', uin 'xxxxxx',
own_uin 'xxxxxx', vpcid 'vpc-xxxxxx', subnetid 'subnet-xxxxx');
CREATE SERVER
```

4. The target instance is a self-built instance with public IP in Tencent Cloud.

```
testdb1=>create server srv_test1 foreign data wrapper mysql_fdw options (host 'xxx.xxx.xxx.xx
x', port '3306', access_type '3', region 'ap-guangzhou', uin 'xxxxxx', own_uin 'xxxxxx');
CREATE SERVER
```

5. The target instance is connected via Tencent Cloud VPN.

```
testdb1=>create server srv_test1 foreign data wrapper mysql_fdw options (host 'xxx.xxx.xxx.xx
x', port '3306', access_type '4', region 'ap-guangzhou', uin 'xxxxxx', own_uin 'xxxxxx', vpngw
id 'xxxxxx');
```

6. The target instance is connected via a self-built VPN.

```
testdb1=>create server srv_test1 foreign data wrapper mysql_fdw options (host 'xxx.xxx.xxx.xx
x', port '3306', access_type '5', region 'ap-guangzhou', uin 'xxxxxx', own_uin 'xxxxxx', vpngw
id 'xxxxxx');
```

7. The target instance is connected via Direct Connect.

```
testdb1=>create server srv_test1 foreign data wrapper mysql_fdw options (host 'xxx.xxx.xxx.xx
x', port '3306', access_type '6', region 'ap-guangzhou', uin 'xxxxxx', own_uin 'xxxxxx', dcgid
'xxxxxx');
CREATE SERVER
```

4. Create a user mapping

```
testdb1=> create user mapping for user1 server srv_test1 options (user 'user2',password 'password
2');
CREATE USER MAPPING
```

5. Create a foreign table

```
testdb1=> create foreign table foreign_table1(id integer) server srv_test1 options(dbname 'testdb
2', table_name 'test_table2');
CREATE FOREIGN TABLE
```

6. Access foreign data

```
testdb1=> select * from foreign_table1;
id
----
1
(1 row)
```

7. References

CREATE SERVER (9.3)

CREATE SERVER (9.5)

Example of Using cos_fdw

cos_fdw can be used to get text data in COS from a TencentDB for PostgreSQL instance.

1. Prerequisites

1. Create test data in the current instance.

```
postgres=>create role user1 with LOGIN CREATEDB PASSWORD 'password1';
postgres=>create database testdb1;
CREATE DATABASE
```

2. Create a bucket named "test1" in the COS Console and upload a text file to "/testdir/test.txt" in it.

2. Create cos_fdw

```
# Create
postgres=> ¥c testdb1
You are now connected to database "testdb1" as user "user1".
testdb1=> create extension cos_fdw;
CREATE EXTENSION
# View
testdb1=> ¥dx
```

```
List of installed extensions
Name | Version | Schema | Description
-------------+---------+------------+------------------------------------------------------------
plpgsql | 1.0 | pg_catalog | PL/pgSQL procedural language
cos_fdw | 1.0 | public | foreign-data wrapper for flat qcloud cos access
(2 rows)
```

3. Create a server

```
# Access the data of the COS bucket `test1` from `testdb1` in the current instance
testdb1=>create server srv_cos foreign data wrapper cos_fdw options(host 'test11-xxxxxx.cos.ap-ch
engdu.myqcloud.com', bucket 'test1', id 'xxxxxx', key 'xxxxxx');
CREATE SERVER
```

4. Create a foreign table

Parameter: `filepath` , which is the relative path to the text file in the bucket.

```
testdb1=> create foreign table test_cos(id integer) server srv_cos options(filepath '/testdir/tes
t.txt');
CREATE FOREIGN TABLE
```

5. Access foreign data

```
testdb1=> select * from test_cos;
id
----
1
(1 row)
```

6. References

COS documentation