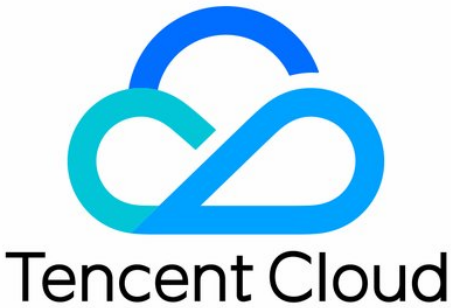
TencentDB for PostgreSQL

Operation Guide

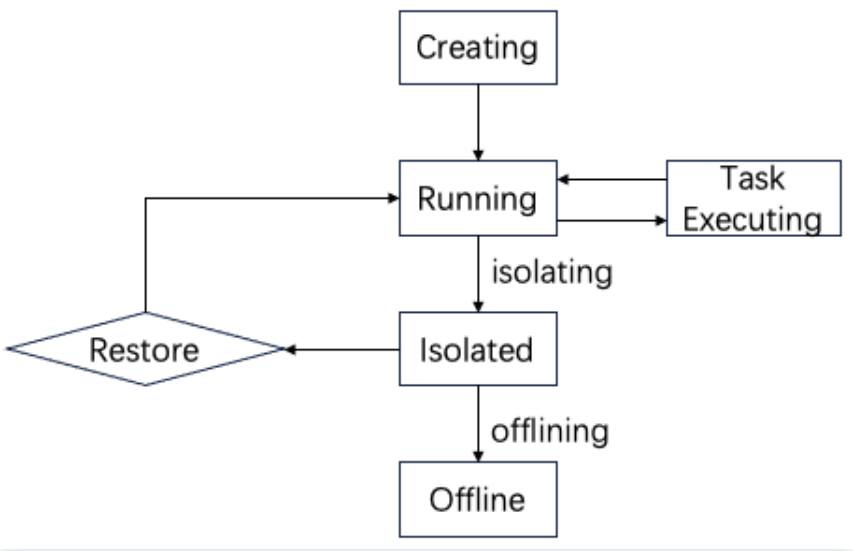


Contents

Operation Guide
Instance management
Instance Lifecycle

Last updated：2023-09-10 08:56:04

TencentDB for PostgreSQL instances have various statuses, and the available operations differ depending on the status. This article primarily focuses on the instance lifecycle and its related aspects.
TencentDB for PostgreSQL instances have the following status:



- In the **Creating** status, the instance is in its initial state. Once the creation is complete, the instance can be used normally.
- The **Running** and **Task Executing** statuses indicate that the instance is operating normally. The Task Executing status signifies that the instance is currently performing some operations, such as configuration changes.
- When a monthly subscribed instance expires, a pay-as-you-go instance user has an outstanding balance, or a user actively terminates an instance, the instance will enter the isolation state and its status will change to **Isolated**. At this point, the instance can be viewed in the recycle bin.
- In the recycle bin, you can perform a **Restore** operation on the instance. After restoration, the instance will return to the Running status.
- Monthly subscribed instances in the recycle bin will be retained for 7 days, while pay-as-you-go instances will be retained for 3 days. After the retention period, the instances will be automatically taken offline. Once offline, the instances will be permanently deleted and cannot be recovered, nor can they be viewed from the console.

Setting Instance Maintenance Time

Last updated: 2023-09-10 08:56:12

Scenario

Maintenance time is crucial for TencentDB for PostgreSQL. To ensure the stability of your TencentDB for PostgreSQL instance, the backend system periodically performs maintenance operations during the maintenance time. We recommend that you set an acceptable maintenance time for your business instance, typically during off-peak hours, to minimize the impact on your business.
In addition, we also recommend you perform operations involving data migration during the maintenance time, such as instance specification adjustment. Currently, the maintenance time is supported by primary and read-only instances.
Taking the database instance specification upgrade as an example, if data migration is involved, a momentary disconnection from the database may occur after the upgrade is completed. By selecting the **switch time** as **during maintenance time**, the instance specification switch will be initiated within the next **maintenance time** after the upgrade is completed. It is important to note that when choosing the switch time as **during maintenance time**, the database specification upgrade will not switch immediately upon completion, but will maintain synchronization until the instance's **maintenance time** initiates the switch, which may extend the overall time required for the instance upgrade.

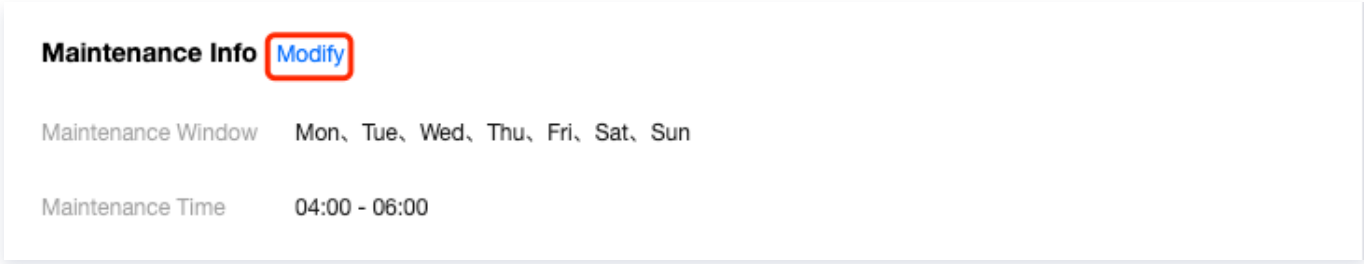
Note

- Before maintenance is carried out for TencentDB for PostgreSQL, notifications will be sent to the contacts configured in your Tencent Cloud account through SMS and email.
- Instance switch will cause a flash disconnection from the database. Please make sure that your business has a reconnection mechanism.

Instructions

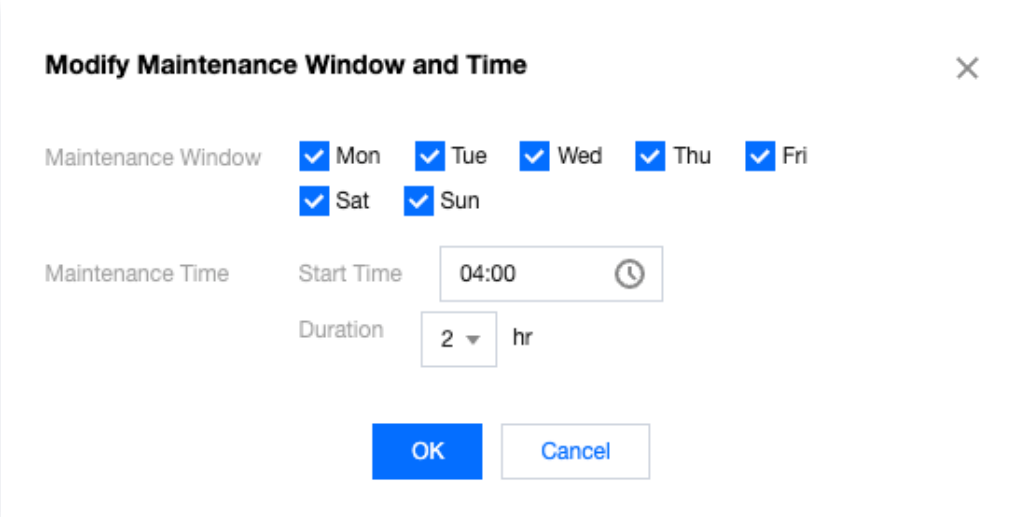
Setting maintenance time

1. Log in to the TencentDB for PostgreSQL console. In the instance list, click an instance ID or **Manage** in the **Operation** column to enter the instance details page.
2. In the **Maintenance Information** section on the details page, click **Modify**.



3. In the pop-up window, select the desired **Maintenance Window** and **Maintenance Time**, then click

Confirm.



Immediate switch

If a task is set to switch during the maintenance time, but due to special circumstances, the switch needs to be performed before the maintenance time, you can click **Immediate Switch** in the **Operation** column.

> **Note**
> Switching now is applicable to operations involving data migration such as instance specification upgrade.

Modifying Instance Configuration

Last updated: 2023-09-10 08:56:21

This document describes how to modify the computing specification and storage capacity of a TencentDB for PostgreSQL instance.

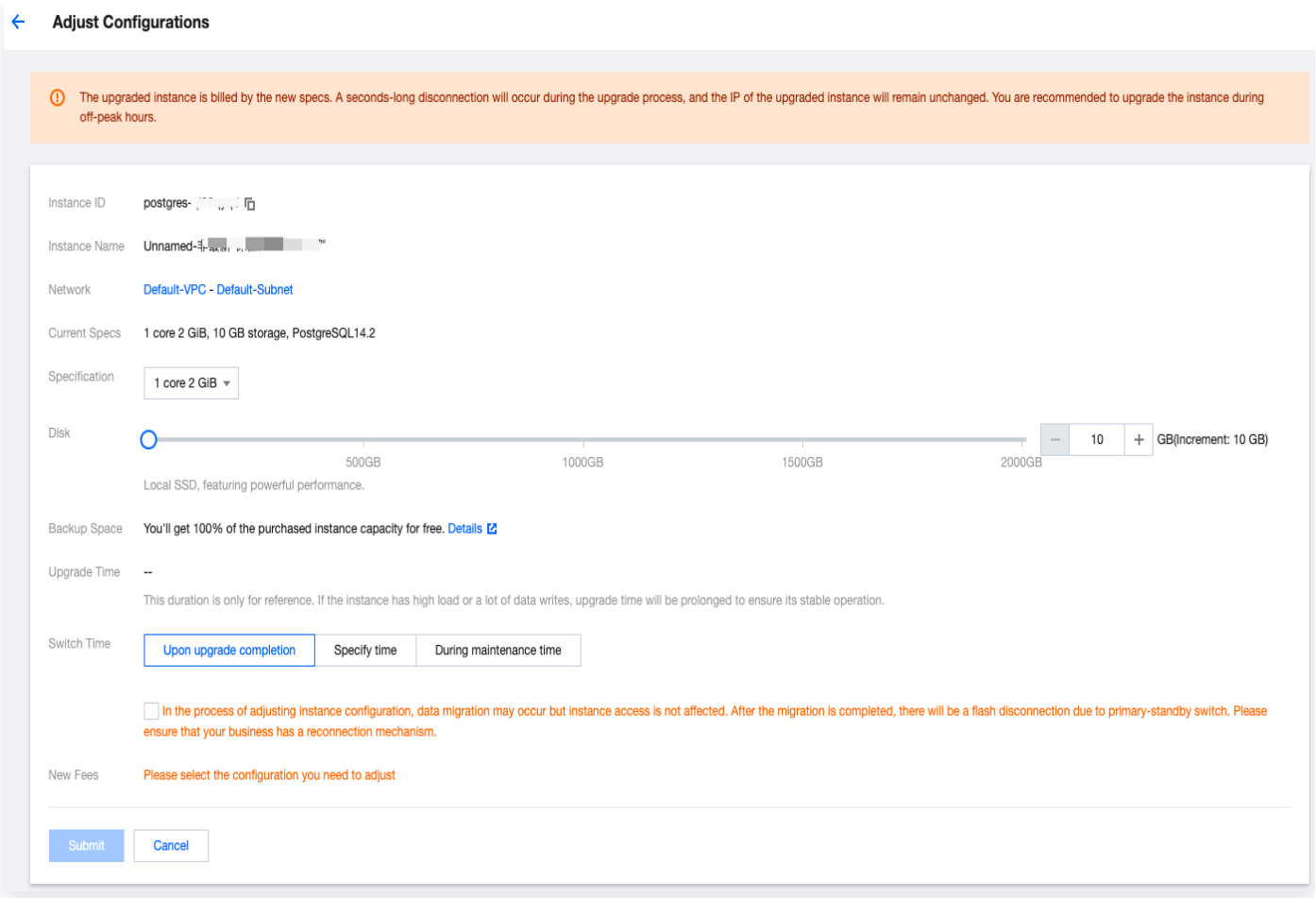
Scenario

When the current performance or storage capacity of an instance cannot meet the needs of business changes, the instance configuration can be adjusted to sustain the business growth.

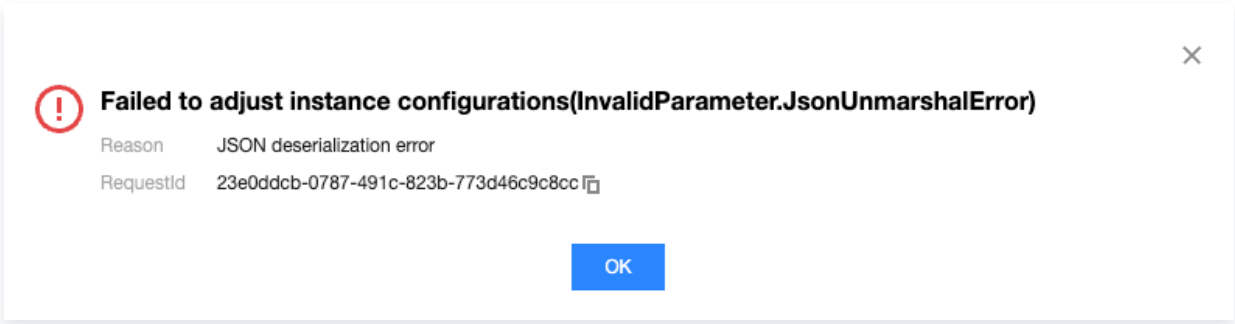
Instructions

1. Log in to the TencentDB for PostgreSQL console. In the instance list, locate the instance to be reconfigured and click **Adjust Configuration** in the **Operation** column.
2. On the pop-up page, select the desired instance configuration and storage capacity. After making your selections, the specific reconfiguration cost will be displayed at the bottom. Confirm the fees and click **Submit**.

> **Note**
> - As instance reconfiguration involves data migration, it will be completed with a primary-secondary switch. This switch may cause a momentary disconnection, which can affect the business access to the database instance to some extent. Therefore, it is generally recommended to schedule the instance switch during off-peak business hours.
> - If you need to control the switch time, select **Specify Time** in the **Switch Time** section. Specify a time range, and after the instance data migration is completed, the system will automatically check if it is within the switch time range. If not, the instance will be in the "Waiting for Switch" state until the switch is completed within the nearest specified time range.
> - The time range is calculated on a daily basis. If you miss the current day's time window, you can only switch during the next day's time window.
> - When the instance is in the **Waiting for switch** status, you can use the **Switch Now** feature in the instance list to complete the reconfiguration immediately.



When adjusting the instance configuration, the system checks whether the runtime values of the instance specification parameters exceed the upper limit of the corresponding range after **modifying the instance specification**. If the values exceed the limit, a corresponding prompt will be displayed, as shown in the figure below:



Isolating Instances

Last updated: 2023-09-10 08:56:36

This document guides you on how to isolate instances using the TencentDB for PostgreSQL console.

Scenario

Based on your business requirements, you can choose to isolate instances at any time. Isolating an instance moves it to the recycle bin and initiates a refund process.

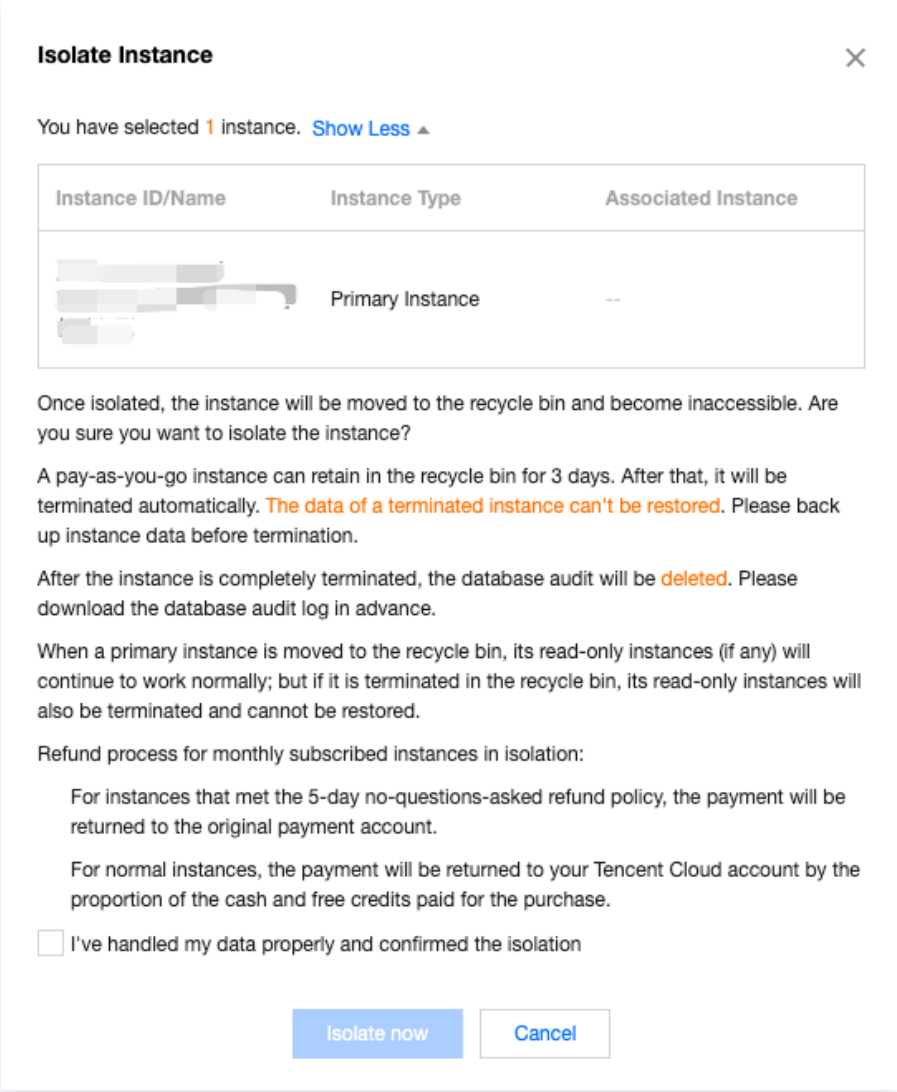
Supports and Limits

- After isolating a monthly subscribed instance, it will automatically be stored in the recycle bin for 7 days, while a pay-as-you-go instance will be stored for 3 days. During this period, the instance status will change to "Isolated," and it will be inaccessible.
  - To use the instance again, you can restore it from the recycle bin.
  - If you do not need to retain the instance, you can perform a termination operation on it within the recycle bin.
- After the instance is terminated, its data cannot be recovered, and its backup files will also be terminated, so the data cannot be restored in the cloud. Store your backup files safely elsewhere in advance.
- When a primary instance has associated read-only instances, isolating the primary instance will not affect the read-only instances. However, destroying the primary instance will result in the destruction of the read-only instances. If you need to continue using the instances, please pay close attention to the expiration information to avoid instance unavailability due to overdue payments and subsequent destruction.
- If a monthly subscribed read-only instance is terminated due to the destruction of its primary instance, a refund will be automatically processed for you.
- If a monthly subscribed instance is restored within 7 days of isolation, fees will continue to be charged during the isolation period. However, if the instance is not needed, no fees will be charged during the isolation period.
- Once a pay-as-you-go instance is isolated, the billing will cease.
- After isolating a monthly subscribed instance, a refund will be automatically processed for you. However, please note the following considerations:
- The 5-day free returns will be refunded to your Tencent Cloud account.
- For normal monthly-subscribed instances, the payment will be returned to your Tencent Cloud account by the proportion of the cash and gift cards paid for the purchase after the termination.
- For orders from promotional reward channels, a refund processing fee of 25% of the actual cash payment amount will be charged. Currently, self-service refunds are not supported for such orders; please initiate a refund request through online support.
- To prevent accidental deletion of instances, Tencent Cloud has implemented key deletion

protection measures. If an instance is configured with Data Encryption, the key will not be unbound immediately after instance isolation or termination. Instead, the key can only be deleted from KMS three days after the instance has been destroyed in the recycle bin.

Instructions

1. Log in to the TencentDB for PostgreSQL console. In the instance list, locate the instance to be isolated, and in the **Operation** column, select **More** > **Isolate Instance**.
2. In the pop-up dialog box, select the confirmation check box and click **Isolate Now**.



3. Upon clicking "Isolate Now," pay-as-you-go instances will be moved to the recycle bin. For monthly subscribed instances, they will be directed to the refund page, where the system calculates the fees based on the usage duration and initiates a refund. Please verify the refund information for accuracy and click "Confirm Refund."
4. Upon completing the refund process, wait for the refund to be successful. Once successful, you will see the "Isolated" instance in the recycle bin.

Restoring Instances

Last updated: 2023-09-10 08:56:44

This document describes how to restore an isolated TencentDB for PostgreSQL instance in the console.

Scenario

If an instance is terminated by mistake, due to overdue payment, or when it expires, you can go to the recycle bin to restore it before it is eliminated.

> **Note**
> - After an instance is restored, it uses the same configurations as before.
> - An instance cannot be terminated, restored and terminated again in a short time.

Instructions

1. Log in to the TencentDB for PostgreSQL console, locate the instance to be restored in the recycle bin list, and click **Restore** in the **Operation** column.
2. In the pop-up window, confirm the renewal and recharging information, then click **OK**.
3. After the restoration is completed, you can see the instance in the instance list.

Restarting Instance

Last updated: 2023-09-10 08:57:02

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.

Supports and Limits

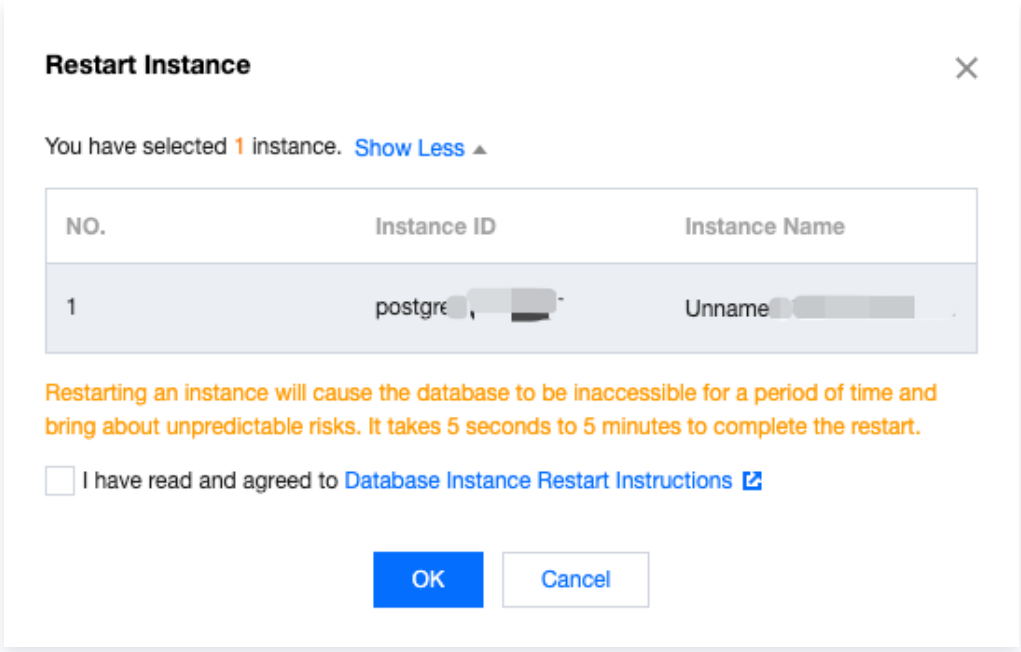
- Please exercise great caution when restarting a database, which plays a vital role in the business. Before the restart, it is recommended to disconnect the database from server and stop writing data.
- Restarting an instance does not change its physical attributes, so the public IP, private IP, and any data stored on the instance will remain unchanged.
- After the restart, reconnection to the database is needed. Make sure your business has a reconnection mechanism.
- Restart the instance during off-peak hours to ensure success and minimize the impact on your business.

Instructions

1. Log in to the TencentDB for PostgreSQL console, select the desired instance from the instance list, and click **More** > **Restart** in the **Operation** column.

   > **Note**
   > - Generally, it takes a few seconds to minutes to restart an instance, during which the instance cannot be accessed and existing connections to it will be closed.
   > - During the restart, if the volume of business writes is too large and there are too many dirty pages, the restart may fail. If the restart fails, the instance will return to its previous state and remain accessible.
   > - Restarting the database may occasionally fail, which is normal. If the restart takes longer than 10 minutes, consider consulting online support.

2. In the pop-up window, after agreeing to the terms, click **OK**



Modifying Data Replication Mode

Last updated: 2023-09-10 08:58:32

Supported Data Replication Modes

Database instance replication refers to the process of synchronizing data by configuring one or more backup databases for a server, distributing PostgreSQL data across multiple systems. TencentDB for PostgreSQL supports the following two data replication methods:

Async replication

- TencentDB for PostgreSQL async replication adopts a one-primary-one-replica architecture.
- After receiving a data update (including INSERT, UPDATE and DELETE operations) request from an application, the source performs the update operation. When the update is completed, the source immediately responds to the application and replicates the data to the replica.
- During the data update process, the source does not need to wait for the replica's response. Therefore, database instances with async replication usually have higher performance (for specific performance, see Test Results). The unavailability of the replica does not affect the source's service provision. However, since the data is not synchronized to the replica in real-time, and if a switch occurs after the source fails when there is a delay in the replica, there is a small probability that data inconsistency may occur.

> **Note:**
> TencentDB for PostgreSQL uses async replication as the default data replication method.

Semi-sync replication

- TencentDB for PostgreSQL semi-sync replication adopts a one-primary-one-replica architecture.
- Upon receiving a data update request (including INSERT, UPDATE, and DELETE operations) from an application, the source immediately replicates the data to the replica after performing the update operation. The replica, after receiving the data and writing it to the WAL (without executing), returns a success message to the source. The source must receive this success message from the replica before it can respond to the application.
- In the event of a replication exception (such as the replica node being unavailable or a network issue affecting data replication), the primary node will temporarily pause (approximately 10 seconds by default in PostgreSQL) its response to the application and downgrade the replication mode to async replication. Once data replication returns to normal, it will resume semi-sync replication.

Degradation Description

Degradation of Faults

If the current PostgreSQL primary-standby replication mode is semi-sync, when a replication exception occurs (such as the replica node being unavailable or a network issue), the primary node will temporarily pause its response to the application (approximately 10 seconds by default in TencentDB for PostgreSQL) and downgrade the replication mode to async to ensure system availability. Once the high-availability system detects that replication has returned to normal, it will restore the primary-standby replication mode to semi-sync replication.

> ⚠ **Note:**
> Degradation to high availability is the default behavior for TencentDB for PostgreSQL. To ensure system high availability, the option to modify this setting is not currently available.

Latency degradation

If you have special requirements, you can enable delay degradation under semi-sync replication. After enabling delay degradation, the TencentDB for PostgreSQL high-availability system will judge the primary-standby replication delay based on the conditions you set. If the delay exceeds the set limit, semi-sync will degrade to async. It is recommended to enable this feature only for businesses that are extremely sensitive to delay.
Degradation conditions are based on the size or time of primary-replica synchronization. Relevant monitoring metrics include replica WAL flush lag (in bytes) and replica WAL flush time lag (in seconds). For more information, refer to Primary-Replica Latency Monitoring Metrics.

Failover Description

When the primary-replica replication mode of an instance is set to **async replication** or **semi-sync replication degraded to async replication**, a primary-replica switch will be triggered if the source fails and cannot be recovered. As data is not synchronized to the replica in real-time, there is a small probability of data inconsistency. TencentDB for PostgreSQL provides configurable failover conditions for flexible settings. By default, the system allows switching when both **primary-replica sync delay of 1024MB** and **primary-replica delay of 10 seconds** conditions are met. It is recommended to adjust these settings only if you have specific business requirements.

Modifying Data Replication Mode

1. Log in to the TencentDB for PostgreSQL console. In the instance list, click an instance ID or **Manage** in the **Operation** column to enter the instance details page.
2. In the **Availability Information** section of the Instance Details, detailed availability information of the instance is displayed.
   2.1 When the data replication mode is set to async, the specific information displayed is as follows:

Info	Note
Data Replication	The data synchronization method between primary and standby instances supports two modes in the current dual-machine high-

Mode	availability (one-primary-one-replica) architecture: **async replication** and **semi-sync replication**.
Instance Availability Status	The current accessibility status of the instance is displayed. When the status is normal, it can accept user requests. If the status is abnormal, it indicates that the instance is currently unable to accept application requests.
Failover Conditions	In case of an unrecoverable failure of the source node, an automatic failover is required, and the replica takes over the service. The system defines failover conditions based on the replication delay size and replication delay time. The default conditions are 1024MB and 10 seconds. Applications can modify the switch conditions based on specific requirements. For detailed switch conditions, please refer to Failover Description.
Primary AZ	Availability zone of the Master node.
Replica AZ	Replica node's availability zone.

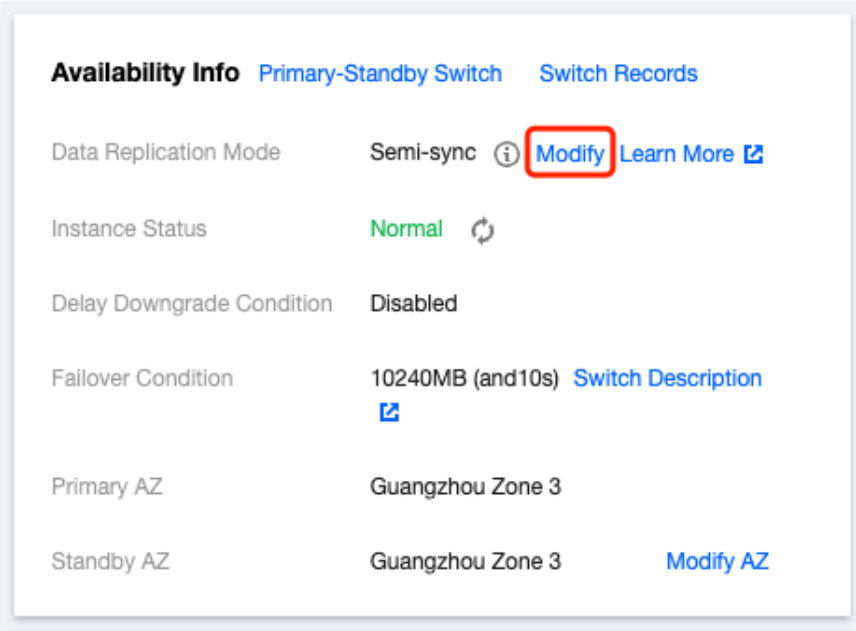
2.2 When the data replication mode is set to semi-sync, the specific information is displayed as follows:

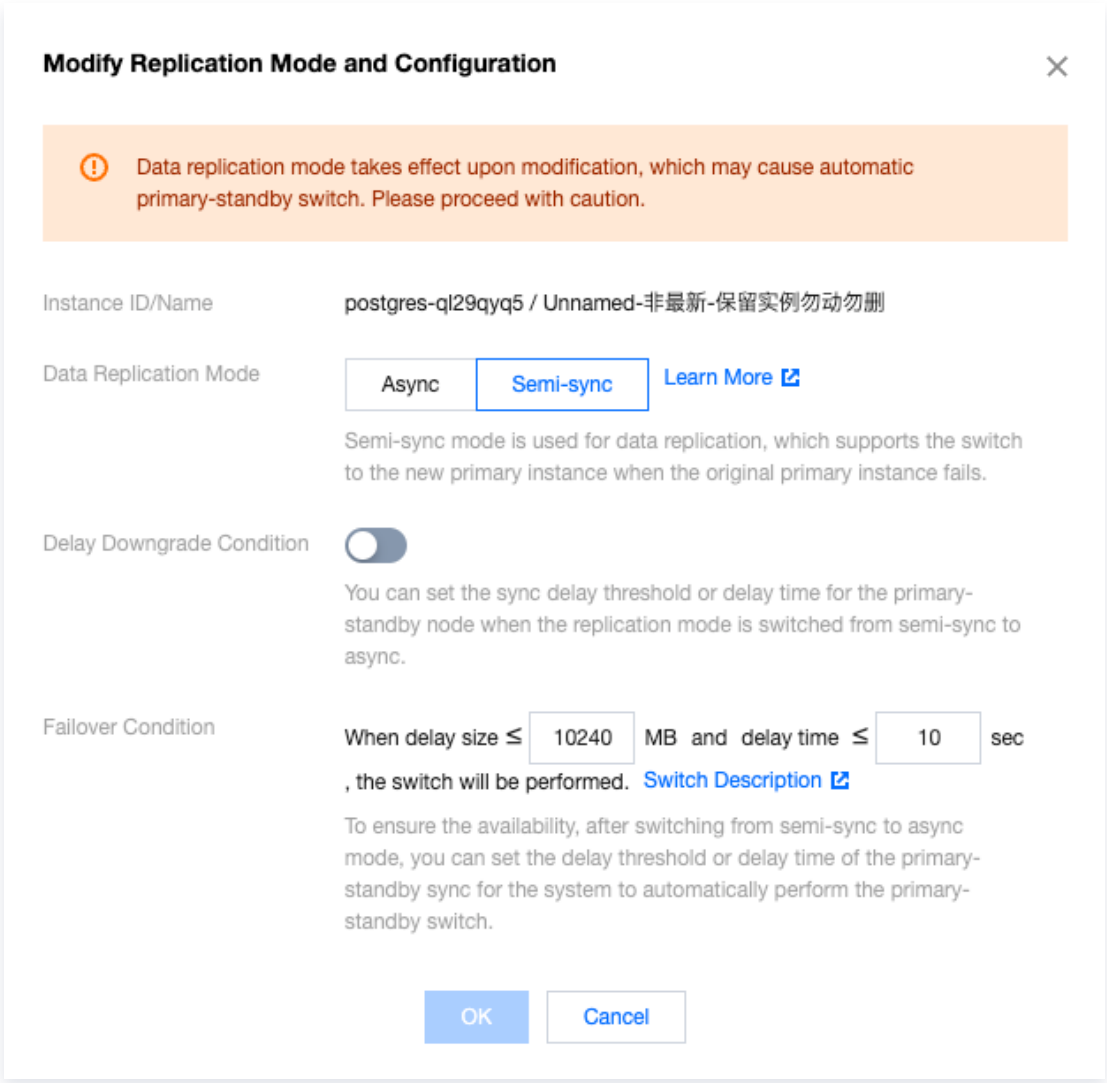
Info	Note
Data Replication Mode	The data synchronization method between primary and standby instances supports two modes in the current dual-machine high-availability (one-primary-one-replica) architecture: **async replication** and **semi-sync replication**.
Instance Availability Status	The current accessibility status of the instance is displayed. When the status is normal, it can accept user requests. If the status is abnormal, it indicates that the instance is currently unable to accept application requests.
Degradation Conditions	When the instance data replication mode is set to semi-sync replication, the system will automatically degrade the primary-standby replication mode to async replication outside the user-defined conditions to ensure system availability. The degradation conditions are based on primary-standby delay size or delay time. For PostgreSQL instances with major version 9, only the primary-standby delay size condition is supported. For more details, please refer to Degradation Description.
Failover Conditions	When the source node fails and cannot be recovered, an automatic failover is required, and the replica takes over the service. The system defines failover conditions based on the primary-replica delay size or time. Applications can modify the switch conditions based on specific requirements. For more information, please refer to Failover Description.

Primary AZ	Availability zone of the Master node.
Replica AZ	Replica node's availability zone.

3. Click **Modify** to change the data replication mode of the current instance.

> **Note:**
> The change in data replication mode takes effect immediately. Modifying the replication mode may cause a primary-replica switch, which can result in a momentary disconnection. Please ensure that your application has a reconnection mechanism.





Switch Instance Master/Replica

Last updated: 2023-09-10 08:58:43

Master/Replica Switch Reasons

The interchange of roles between the Master and Slave nodes of an instance is referred to as a master/replica switch. After the switch, the instance address remains unchanged, and the application automatically connects to the new Master node, ensuring high availability of the instance. The main reasons for a master/replica switch are as follows:

Failover

An automatic master/replica switch initiated when the system detects an abnormality in the instance, rendering it unusable. For specific switch conditions, please refer to Failover Condition Description.

Manual Switching

A master/replica switch initiated manually by application maintenance personnel or authorized Tencent Cloud technical experts. Manual switching includes both regular delay switching and forced switching that exceeds the master/replica delay.

Forced Switching

When the primary/replica replication mode of an instance is asynchronous replication or semi-synchronous replication downgraded to asynchronous replication, a master/replica switch will be triggered if the Master fails and cannot be recovered. As data is not synchronized to the Slave in real-time, there is a small probability of data inconsistency. The current system has default switch conditions set, but you can also make special settings based on your business needs. Therefore, a switch is only allowed when the switch conditions are met. To accommodate emergency switches, the system provides a forced switching capability.

> ⚠ **Note:**
> To prevent switching conditions from changing over time, an immediate switch must be executed when performing a forced switch.

Master/Replica Switch Impact

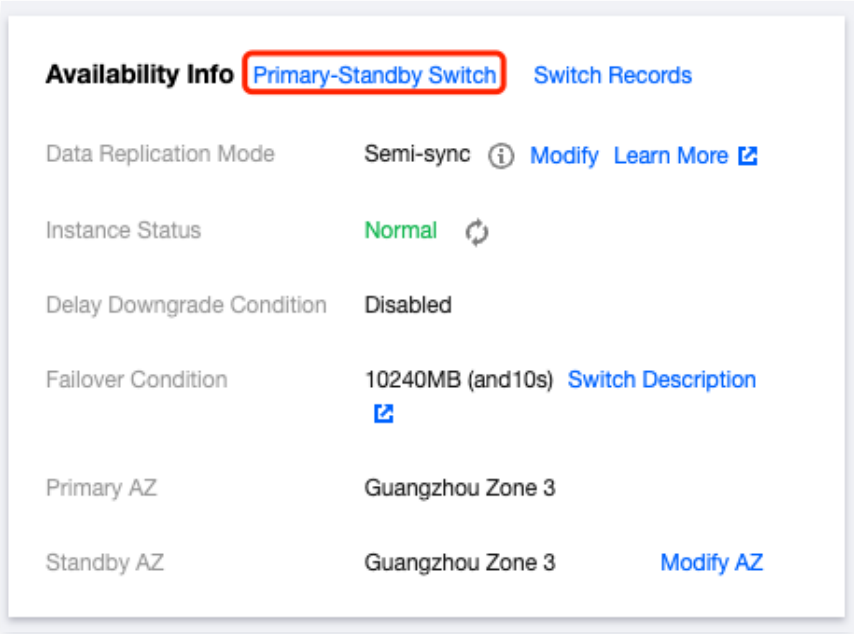
- During the master/replica switch, a momentary disconnection will occur. Please ensure that your application has a reconnection mechanism.
- If the primary instance has read-only instances attached, there will be a minute-level delay in the read-only instances after the master/replica switch.

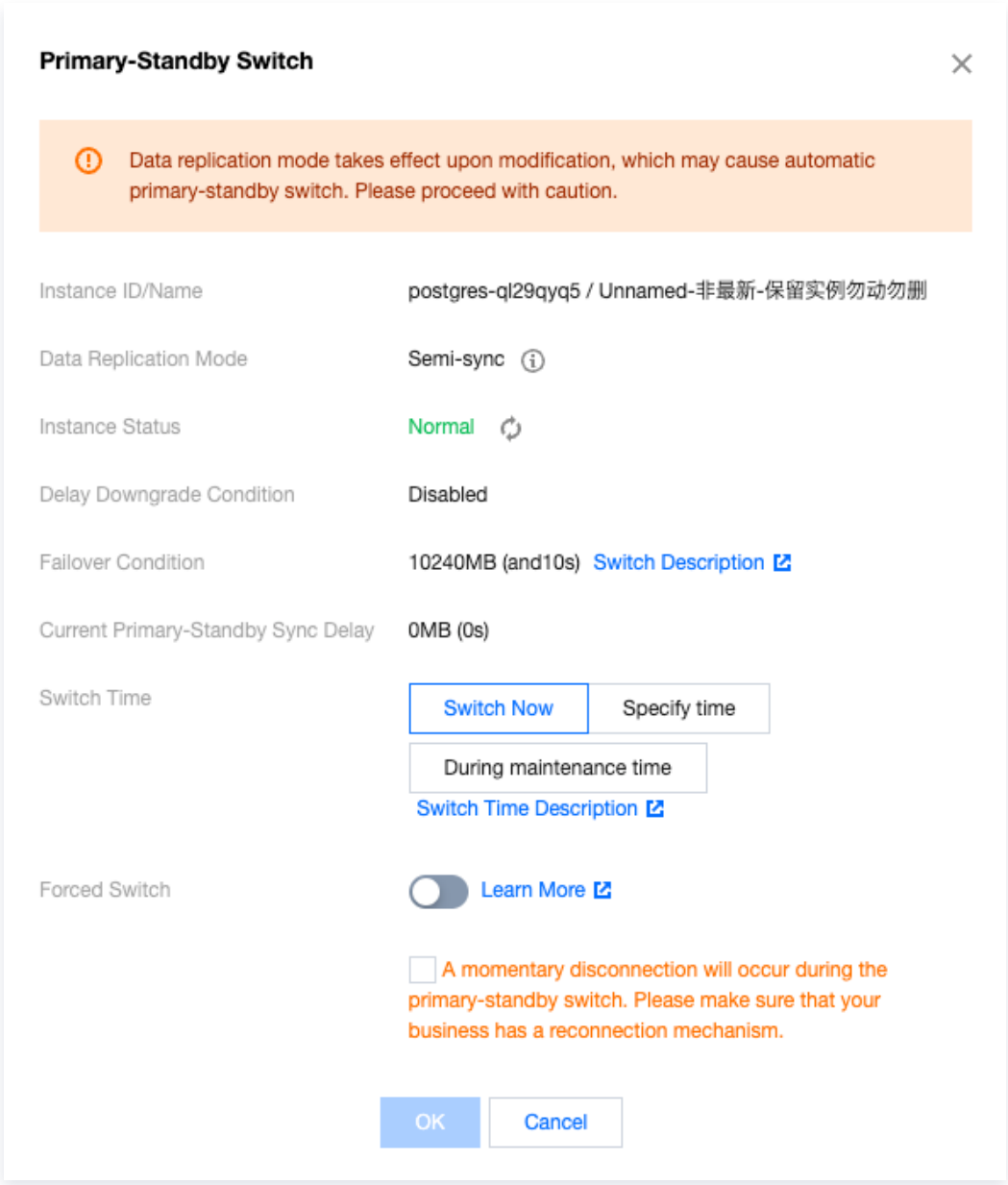
Manually Switch Instance

1. Log in to the TencentDB for PostgreSQL console. In the instance list, click an **instance ID** or

Manage in the **Operation** column to enter the instance details page.

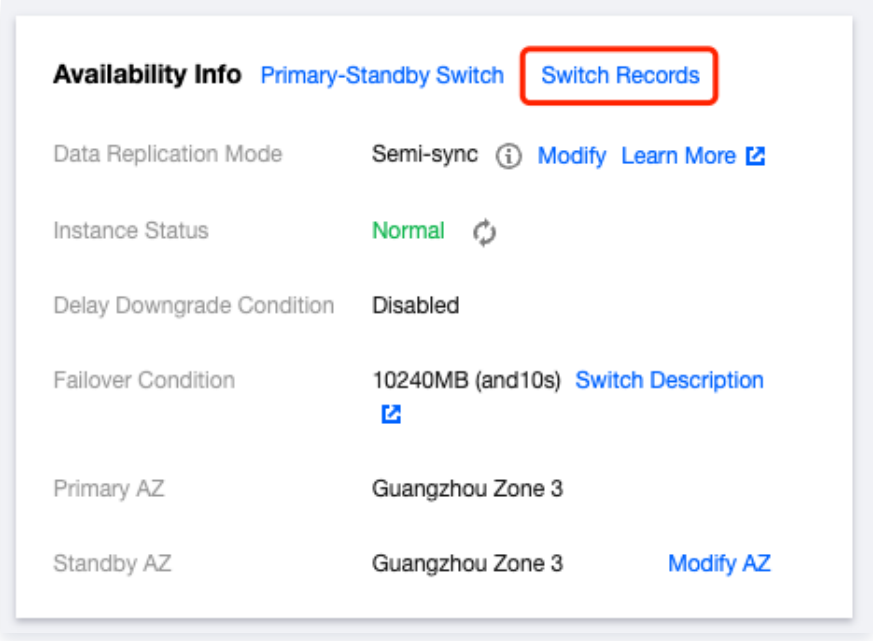
2. In the **Availability Info** section of the Instance Details, click **Master/Replica Switch**.





Viewing Switch Records

1. Log in to the TencentDB for PostgreSQL console. In the instance list, click an **instance ID** or **Manage** in the **Operation** column to enter the instance details page.
2. In the **Availability Information** section of the Instance Details, click **Switch Records**. The system will retain switch records for one year.



Primary–Secondary Delay Monitoring Metric

TencentDB for PostgreSQL provides detailed monitoring information to help you view the synchronization latency between the master and replica nodes. The specific monitoring metrics are as follows:

Meaning	Metric Description in Chinese
Replica WAL Flush Lag (Bytes)	The size difference between the Replica's stored LSN and the Master instance's current LSN. For the Master instance, this metric reflects the amount of data loss during a failover.
Replica WAL Flush Lag (seconds)	The time difference between sending logs from the primary instance to the replica instance and the replica instance receiving and writing the logs to disk. For the primary instance, this metric reflects the amount of data loss during a failover. This metric is available for instance versions 10.x and above.
Master/Replica Data Synchronization Delay (Bytes)	The size difference between the replica's replay LSN and the current LSN of the primary instance. For the primary instance, this metric reflects the Recovery Time Objective (RTO) during failover. For read-only instances, this metric indicates the data latency.
Master/Replica Data Synchronization Delay Time (seconds)	The time difference between logs being sent from the primary database to the replica database and the replica database receiving and replaying the logs. This metric is available for instance versions 10.x and above.
Difference in Log Send and	The size difference between the log sent from the primary

Replay Positions (Bytes) for Replica Nodes	database to the replica database and the completion of the replica's log playback primarily reflects the speed of log application on the replica. This metric mainly allows for the assessment of the replica's performance and the speed of network transmission. Read-only instances do not have this metric.

Read-only instance
Read-Only Instance Overview

Last updated: 2023-09-10 09:01:11

TencentDB for PostgreSQL allows you to create one or more read-only instances, which are suitable for read/write separation and one-primary-multi-standby application scenarios and capable of greatly enhancing the read load capacity of your database.
Currently, a unified read/write separation address, which automatically separates read and write requests, is not supported. Read-only instances can be accessed via individual IP and PORT, or they can be assigned to a read-only instance group for unified load balancing.

Note

- For read-only instance costs, see Pricing.
- You can create one read-only instance at a time, with a maximum of six read-only instances allowed.
- Configure two or more read-only instances to prevent business access failure due to a single point of failure in the read-only instance.
- Read-only instances do not have high availability capabilities. It is recommended to use read-only instance groups. A read-only group with only one read-only instance may pose a single point of risk, and this group will not be included in the overall availability calculation of the TencentDB for PostgreSQL service. Individual read-only instances do not provide availability SLA guarantees. It is recommended to purchase at least two read-only instances in a read-only group to ensure the group's availability.

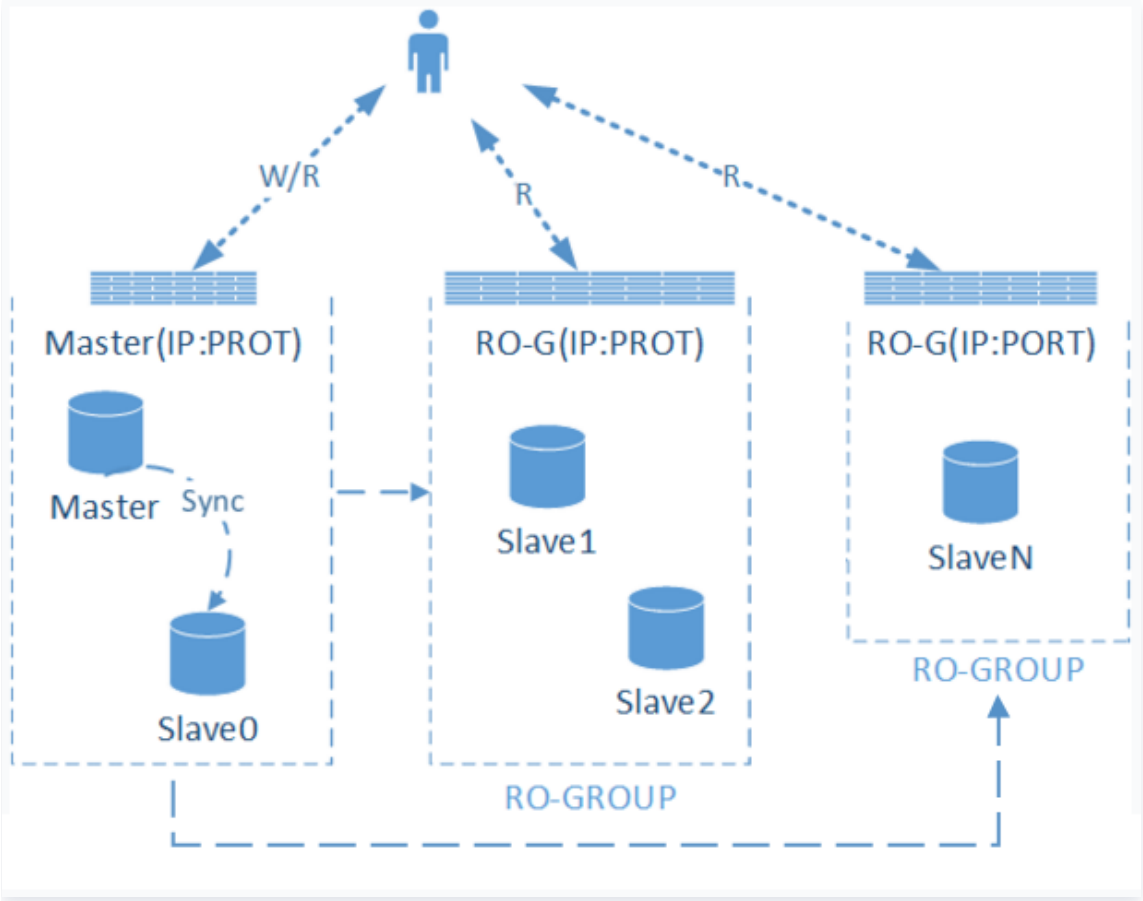
Concepts

- Read-only instance group: A read-only instance (RO) group with load balancing capabilities. If there are multiple read-only instances within the group, user read requests can be evenly distributed among them. Additionally, the RO group provides an IP and PORT for external access to the database.
- Read-Only Instance: A single-node (without slave) instance that supports read requests. A read-only instance cannot exist independently and must be associated with a primary instance.

Basic Infrastructure

Read-only instances employ the PostgreSQL streaming replication mechanism to synchronize changes from the primary instance (source database) to all read-only instances. Each read-only instance uses a single-node (without standby) architecture. If a read-only instance fails, we will

continuously attempt to recover it. For higher availability, you can opt for an RO group.



Feature limits

- The minimum disk capacity for a read-only instance must be greater than or equal to the storage capacity used by the primary instance.
- Up to 6 read-only instances can be created for a primary instance.
- Backup and rollback features are not supported.
- Data cannot be migrated to read-only instances.
- Databases can be neither created in nor deleted from read-only replicas.
- Operations including account creation/deletion/authorization and account name/password modification are not supported.

Supports and Limits

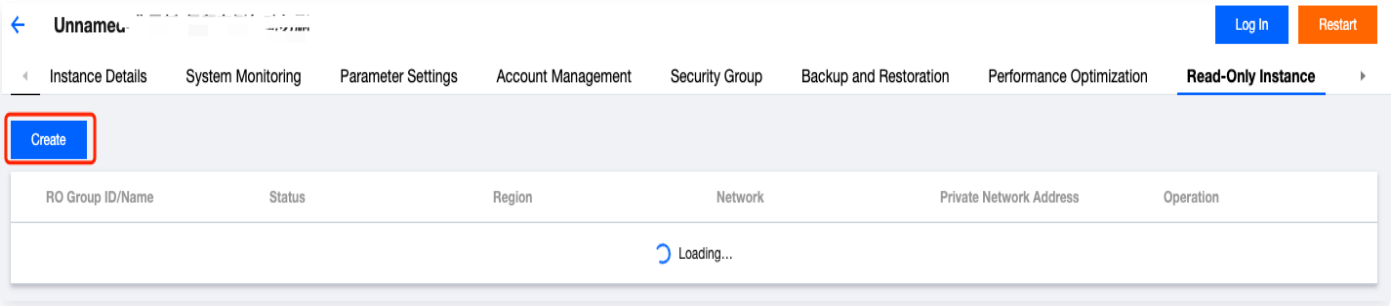
- There is no need to maintain accounts or databases for read-only instances, which are synchronized with those of the primary instance.
- Due to data synchronization delays, there may be minor inconsistencies among multiple read-only instances. The synchronization latency between each read-only instance and the primary instance can be viewed in the console, and alarms can be set through the Tencent Cloud Observability Platform.
- The specification of a read-only instance can be different from that of the source instance, which

makes it easier for you to upgrade the read-only instance based on the load. It is recommended to ensure that all RO instances in the same RO group have the same specification.

- If the primary instance is written so frequently that the automatic log cleanup threshold is exceeded, logs will be automatically deleted. If the standby instance hasn't obtained the deleted logs yet, the primary-standby replication will be disconnected, and the read-only instance will be automatically recreated and become inaccessible.
- Read-only instances don't have high availability. We recommend you use an RO group and configure at least two read-only instances to avoid business access failures caused by single points of failures.

Instructions

1. Log in to the TencentDB for PostgreSQL console. In the instance list, click an **instance ID** or **Manage** in the **Operation** column to enter the instance management page.
2. In the Instance Details page, click **Add Read-Only Instance** in the **Instance Architecture Diagram**, or click **Create** on the Read-Only Instance page to enter the purchase page.



3. On the purchase page, select the appropriate configuration for the read-only instance, verify the details, and click **Buy Now**.

> **Note**
> To synchronize the expiration dates of read-only instances and primary instances, you can set a unified expiration date in the Renewal Management Console. For detailed steps, please refer to Unified Expiration Date.

- **Specify RO Group**: Options include not assigning an RO group, creating a new RO group, or using an existing RO group.
  - New RO group: If multiple instances are purchased at once, they will all be assigned to this RO group. The RO group only supports system-assigned read weights, and the traffic distribution among all read-only instances in the group will be automatically allocated based on their read weight values. See Creating a Read-Only Instance RO Group for more information.
  - **Existing RO Group**: specify an existing RO group. If multiple read-only instances are purchased at a time, all of them will be assigned to the RO group.
- **Remove Delayed Instances**: After enabling the delay removal feature, you can determine whether to remove read-only instances from the RO group based on the configured delay size. The delay

size refers to the log size difference between the primary instance and the read-only instance during data synchronization, with a configuration granularity of MB.

- **Availability Zone**: You can choose from all AZs within the same region as the primary instance.

4. After completing the purchase, return to the instance list. Once the instance status changes to **Running**, it can be used normally.

Removal Policy and Load Balancing

Last updated: 2023-09-10 09:01:47

TencentDB for PostgreSQL allows users to create multiple read-only instances and form read-only instance RO groups, which are suitable for read/write separation and master-slave application scenarios, significantly enhancing the read load capacity of user databases. This article mainly introduces some management features of RO groups.

Rebalancing Traffic

- If load rebalancing is disabled, modifying weight will take effect only for new loads but not affect the read-only replicas accessed by existing persistent connections or cause short disconnection from the database.
- When load rebalancing is enabled, the read weight value of read-only instances in the RO group changes after an upgrade. Once the read-only instances are upgraded, all connections in the RO group will be disconnected and re-established. All new connections will be load balanced according to the configured weights.

  If you are not satisfied with the current distribution of connection counts among instances in the RO group, you can manually rebalance the connections. Log in to the Console, click on the instance ID to enter the management page, and perform **Rebalance** for the instance in the **Read-Only Instances** tab.

> **Note**
> Ensure that your business has an automatic reconnection mechanism. If there is no automatic reconnection mechanism, do not enable automatic rebalancing or manual rebalancing.

Fault Eviction

When a read-only instance in a RO group becomes inaccessible due to an unexpected error, the RO group automatically removes the read-only instance. This rule is a default rule.

Remove Delayed RO Instances

After enabling the delay removal feature, the read-only instances can be removed from the RO group based on the user-configured delay size. The delay size refers to the log size difference between the source instance and the read-only instance during data synchronization, with a configuration granularity of MB.

Assign Read Weight

In a read-only instance RO group, traffic distribution among all read-only instances is automatically allocated based on their read weight values, achieving load balancing capabilities and avoiding the

increased management difficulty of multiple read-only instance addresses and IP addresses. RO groups only support system-assigned weights. The read weight value list for different specifications of read-only instances is as follows:

Instance Specification	Weight
2GB memory	1
4GB memory	2
8GB memory	2
12GB memory	4
16GB memory	4
24GB memory	8
32GB memory	8
48GB memory	10
64GB memory	12
96GB memory	14
128GB memory	16
240GB memory	26
480GB memory	50

Related Actions

- TencentDB for PostgreSQL allows users to create one or more read-only instances. For more information, see Create Read-Only Instance.
- TencentDB for PostgreSQL allows users to create multiple read-only instances and form read-only instance RO groups. For more information, see Managing Read-Only Instance RO Groups.

Account Management
Database Privilege Overview

Last updated: 2023-09-10 09:02:00

Account Privilege System

The PostgreSQL permission management model is a typical implementation of RBAC (Role-Based Access Control). It manages users, roles, and permissions through this model.
In PostgreSQL, the concepts of users and roles are almost the same. The only difference is that a user has the login privilege, while a role has the nologin privilege.
PostgreSQL permissions can be divided into two categories: "system permissions" and "database object permissions." PostgreSQL manages permissions through roles, which can have both system and database object permissions. Roles can also serve as a collection of permissions granted to other roles or users. Database management can be achieved by granting appropriate system and object permissions to roles/users.

System Permissions

System permissions refer to the privileges that allow specific database operations to be performed. In PostgreSQL, system permissions are managed using two methods: "Role Attributes" and "Default Roles."

Role Attributes

Role attributes can be specified during CREATE ROLE or modified using ALTER ROLE. Role attributes are stored in the pg_authid system table.
The syntax for CREATE ROLE is as follows:

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

```
| USER role_name [, ...]
| SYSID uid
```

The superuser role attribute can bypass all permission checks and perform any operation in the database, holding the highest authority. The superuser privilege is similar to the root privilege in the Linux operating system.

> **Note**
> In accordance with security requirements, TencentDB for PostgreSQL has disabled the use of superuser privileges. However, since some operations require superuser access, TencentDB for PostgreSQL provides the tencentdb_superuser role. Please refer to User and Permission Operations for more information.

Default Role

PostgreSQL provides a set of default roles that grant access to certain privileged features and information. Administrators can grant these roles to other users or roles, allowing them to access the specified features and information. The following table lists the default roles supported in PostgreSQL 11.

Role	Allowed operation
pg_execute_server_program	Allow the execution of corresponding operating system statements through functions or features that can perform partial operation statements.
pg_monitor	The permission to read and execute functions or views related to monitoring. This role includes pg_read_all_settings, pg_read_all_stats, and pg_stat_scan_tables.
pg_read_all_settings	Read all configuration information, including some information that only superusers can access.
pg_read_all_stats	Read all pg_stat_* views and utilize various statistics-related extensions, including some information visible only to superusers.
pg_read_server_files	Allows reading of certain authorized files on the server where the database is located, through copy or file access functions.
pg_signal_backend	You can initiate or terminate a session by sending commands to the backend.
pg_stat_scan_tables	Execute monitoring functions that may require long-term table locking (ACCESS SHARE lock).
pg_write_server_files	Allows writing to some authorized files on the server where the database is located, through copy or file access functions.

public	Public role is a hidden role that represents everyone. If a permission is granted to the public role, then all roles have that permission. Some objects have default permissions granted to the public role.

Database Object Permissions

Database object permissions are controlled using ACL (Access Control List). The table below lists all database object permissions and their abbreviations in PostgreSQL.

Permission	Abbreviation	Supported Objects
SELECT	r (“read ”)	LARGE OBJECT、SEQUENCE、TABLE (and table-like objects)、table column
INSERT	a (“appe nd”)	TABLE、table column
UPDATE	w (“write ”)	LARGE OBJECT、SEQUENCE、TABLE、table column
DELETE	d	TABLE
TRUNCATE	D	TABLE
REFERENCES	x	TABLE、table column
TRIGGER	t	TABLE
CREATE	C	DATABASE、SCHEMA、TABLESPACE
CONNECT	c	DATABASE
TEMPORARY	T	DATABASE
EXECUTE	X	FUNCTION、PROCEDURE
USAGE	U	DOMAIN、FOREIGN DATA WRAPPER、FOREIGN SERVER、LANGUAGE、SCHEMA、SEQUENCE、TYPE

The following table lists the permissions owned by a type of objects and the psql command to query the permissions:

object type	All Permissio	Default Permissions for the Public Role	psql command to view permissions

	ns		
DATABASE	CTc	Tc	\l
DOMAIN	U	U	\dD+
FUNCTION or PROCEDURE	X	X	\df+
FOREIGN DATA WRAPPER	U	none	\dew+
FOREIGN SERVER	U	none	\des+
LANGUAGE	U	U	\dL+
LARGE OBJECT	rw	none	–
SCHEMA	UC	none	\dn+
SEQUENCE	rwU	none	\dp
TABLE (and table-like objects)	arwdDxt	none	\dp
Table column	arwx	none	\dp
TABLESPACE	C	none	\db+
TYPE	U	U	\dT+

In PostgreSQL, an aclitem represents a specific permission on a database object. For databases and schemas, aclitem is stored in pg_database.datacl and pg_namespace.nspacl. For other database objects like tables and views, a list of aclitem is saved in pg_class.relacl. For column-level permissions, aclitem is stored in pg_attribute.attacl.
For instance, normal_user=a*r/test1 indicates that the user 'normal_user' has INSERT and SELECT permissions for the current database object. The INSERT permission comes with a 'with grant option', which means it can be granted to other users. The final '/test1' indicates that this aclitem permission was granted by 'test1'.

```
postgres=# \dp
                        Access privileges
 Schema | Name | Type  |  Access privileges   | Column privileges | Policies
--------+------+-------+----------------------+-------------------+----------
 public | t1   | table | test1=arwdDxt/test1 |                   |
(1 rows)
postgres=# grant select on t1 to normal_user;
GRANT
postgres=# grant insert on t1 to normal_user with grant option;
GRANT
```



```
postgres=# grant update on t1 to public;
GRANT
postgres=# grant select (a) on t1 to test2;
GRANT
postgres=# \dp
                        Access privileges
 Schema | Name | Type  |  Access privileges    | Column privileges | Policies
--------+------+-------+-----------------------+-------------------+----------
 public | t1   | table | test1=arwdDxt/test1  +| a:               +|
        |      |       | normal_user=a*r/test1+|  test2=r/test1    |
        |      |       | =w/test1              |                   |
(1 rows)
-- Where, "=w/test1" specifies that test1 grants public the UPDATE privilege.
```

Users and Permission Operations

Last updated: 2023-09-10 09:02:08

TencentDB Default Role

TencentDB for PostgreSQL does not grant users access to the superuser role attribute, pg_execute_server_program, pg_read_server_files, or pg_write_server_files roles. However, since some operations require superuser privileges, TencentDB for PostgreSQL offers the pg_tencentdb_superuser role as a substitute for the superuser role.

pg_tencentdb_superuser Role

This role supports both system and database object permissions. The detailed permissions are listed in the table below:

System Permissions

Permission	Description
CREATEDB	Can create databases.
BYPASSRLS	Can bypass all row-level security policy checks.
REPLICATION	By default, the role has replication permissions and can also grant replication permissions to other users.
CREATEROLE	This role has permissions similar to the CREATEROLE in the community edition, but pg_read_server_files, pg_write_server_files, and pg_execute_server_program have been removed from the createrole permissions.

Object Permissions

Objects	Description
database	By default, it has permissions for all databases owned by non-superusers.
schema	By default, it has permissions for all schemas owned by non-superusers.
table/sequence	By default, this role has permissions for all tables and sequences owned by non-superusers.
function	By default, it has permissions for all functions owned by non-superusers.
language	No special permissions.
tablespace	No special permissions.
FDW / foreign	By default, it has permissions for all FDWs/foreign servers with owners who

server	are not superusers.
TYPE	By default, it has permissions for all types owned by non-superusers.

Other operations

- Pub/Sub: the tencentdb_superuser role can implement the pub/sub messaging paradigm, create a publication for all tables, and create slots.
- Extensions: Can create all supported extensions. When creating an extension, the current pg_tencentdb_superuser will temporarily be elevated to superuser status, bypassing all permission checks.
- The load_file permission maintains its default behavior, allowing only the loading of .so files from the /lib/plugins/ directory.
- The tencentdb_superuser role can use the `pgstat_get_backend_current_activity` function to view deadlock details, so that users can easily troubleshoot deadlocks themselves.
- The use of the `pg_signal_backend` function is restricted, and processes of the `pg_tencentdb_superuser` role can only be killed by itself.
- Allow checkpoint. The kernel minor version must be equal to or higher than V10.17_r1.8, V11.12_r1.7, V12.7_r1.8, V13.3_r1.6, V14.2_r1.5, and major version equal to or higher than V15.
- Allows creation and modification of event_triggers with the owner set to tencentdb_superuser. The kernel minor version required for creating and modifying event_triggers must be equal to or higher than V10.17_r1.8, V11.12_r1.7, V12.7_r1.8, V13.3_r1.6, V14.2_r1.5, and major version equal to or higher than V15.

Permission Operations

For more information, see the official documents in the PostgreSQL community:

- Create users, for more details, refer to the official documentation:

```
CREATE USER name [ [ WITH ] option [ ... ] ]

where option can be:

    SUPERUSER | NOSUPERUSER
  | CREATEDB | NOCREATEDB
  | CREATEROLE | NOCREATEROLE
  | INHERIT | NOINHERIT
  | LOGIN | NOLOGIN
  | REPLICATION | NOREPLICATION
  | BYPASSRLS | NOBYPASSRLS
  | CONNECTION LIMIT connlimit
  | [ ENCRYPTED ] PASSWORD 'password' | PASSWORD NULL
  | VALID UNTIL 'timestamp'
  | IN ROLE role_name [, ...]
  | IN GROUP role_name [, ...]
```

```
  | ROLE role_name [, ...]
  | ADMIN role_name [, ...]
  | USER role_name [, ...]
  | SYSID uid
```

- Create Role, for more details, refer to the official documentation:

```
CREATE ROLE name [ [ WITH ] option [ ... ] ]

where option can be:

    SUPERUSER | NOSUPERUSER
  | CREATEDB | NOCREATEDB
  | CREATEROLE | NOCREATEROLE
  | INHERIT | NOINHERIT
  | LOGIN | NOLOGIN
  | REPLICATION | NOREPLICATION
  | BYPASSRLS | NOBYPASSRLS
  | CONNECTION LIMIT connlimit
  | [ ENCRYPTED ] PASSWORD 'password' | PASSWORD NULL
  | VALID UNTIL 'timestamp'
  | IN ROLE role_name [, ...]
  | IN GROUP role_name [, ...]
  | ROLE role_name [, ...]
  | ADMIN role_name [, ...]
  | USER role_name [, ...]
  | SYSID uid
```

- Modify role attributes, for more details, refer to the Official Document:

```
ALTER ROLE role_specification [ WITH ] option [ ... ]

where option can be:

    SUPERUSER | NOSUPERUSER
  | CREATEDB | NOCREATEDB
  | CREATEROLE | NOCREATEROLE
  | INHERIT | NOINHERIT
  | LOGIN | NOLOGIN
  | REPLICATION | NOREPLICATION
  | BYPASSRLS | NOBYPASSRLS
  | CONNECTION LIMIT connlimit
  | [ ENCRYPTED ] PASSWORD 'password' | PASSWORD NULL
  | VALID UNTIL 'timestamp'
```

- Grant object permissions to roles, refer to the official documentation for details:

```
Syntax example
GRANT <permission> on <object> to <role>;
```

- Revoking object permissions for roles, for more details, please refer to the official documentation:

```
Syntax example
REVOKE <permission> ON <object> FROM <role>;
```

- Grant a role to another role:

```
Syntax example
GRANT <role_name> to <other_role>;
```

Parameter management
Parameter Value Limits

Last updated: 2023-09-10 09:03:04

Due to factors such as resource specifications, modifications to certain instance parameters may be subject to limitations. This article primarily discusses parameters that are constrained by instance specifications and their respective constraint values.

Supports and Limits

Setting the following parameters will use system resources. To prevent the database use from being affected by parameter values, configure them as needed.

> **Note**
> Some are key parameters, and changing them may affect your use or the normal running of the instance. Therefore, exercise caution when modifying key parameters.

Limits on Parameter Values Restricted by Specification

Specification\Parameter	max_replication_slots	max_wal_senders	max_worker_processes	max_logical_replication_workers	max_parallel_workers	max_connections
1C2GiB	[10-100]	[27-150]	[4-300]	[4-150]	[0-2]	[100-2048]
2C4GiB	[10-100]	[27-150]	[4-300]	[4-150]	[0-4]	[100-2048]
2C6GiB	[10-150]	[27-200]	[4-400]	[4-200]	[0-8]	[100-2048]
4C8GiB	[10-150]	[27-200]	[4-400]	[4-200]	[0-8]	[100-2048]
4C16GiB	[10-150]	[27-200]	[4-400]	[4-200]	[0-8]	[100-2048]
6C24GiB	[10-200]	[27-250]	[4-500]	[4-250]	[0-12]	[100-2048]
8C32GiB	[10-200]	[27-250]	[4-500]	[4-250]	[0-16]	[100-2048]
8C48GiB	[10-200]	[27-	[4-500]	[4-250]	[0-16]	[100-

		250]				2500]
12C64GiB	[10–400]	[27–450]	[4–900]	[4––450]	[0–24]	[100–3000]
16C96GiB	[10–400]	[27–450]	[4–900]	[4––450]	[0–32]	[100–5000]
20C128GiB	[10–500]	[27–450]	[4–900]	[4––450]	[0–40]	[100–10000]
28C240GiB	[10–600]	[27–650]	[4–1300]	[4––650]	[0–56]	[100–13000]
48C480GiB	[10–600]	[27–650]	[4–1300]	[4––650]	[0–96]	[100–22000]

Parameter Template

Last updated: 2023-09-10 09:03:13

Using the TencentDB for PostgreSQL console, you can create custom parameter templates to configure parameters in batches according to your specific business requirements.
You can utilize a database parameter template to manage the configuration of your database engine. A parameter template serves as a container for engine configuration values, which can be applied to one or more database instances.
The parameter template supports the following features. Users can log in to the PostgreSQL Console and select the **Parameter Template** page from the left navigation to view the parameters:

- Create custom templates to generate tailored parameter optimization solutions.
- Import and generate templates from PostgreSQL configuration files `**.conf`.
- Save parameter configurations as templates.
- Import parameters from templates to apply to one or more instances.
- Compare two parameter templates.

Supports and Limits

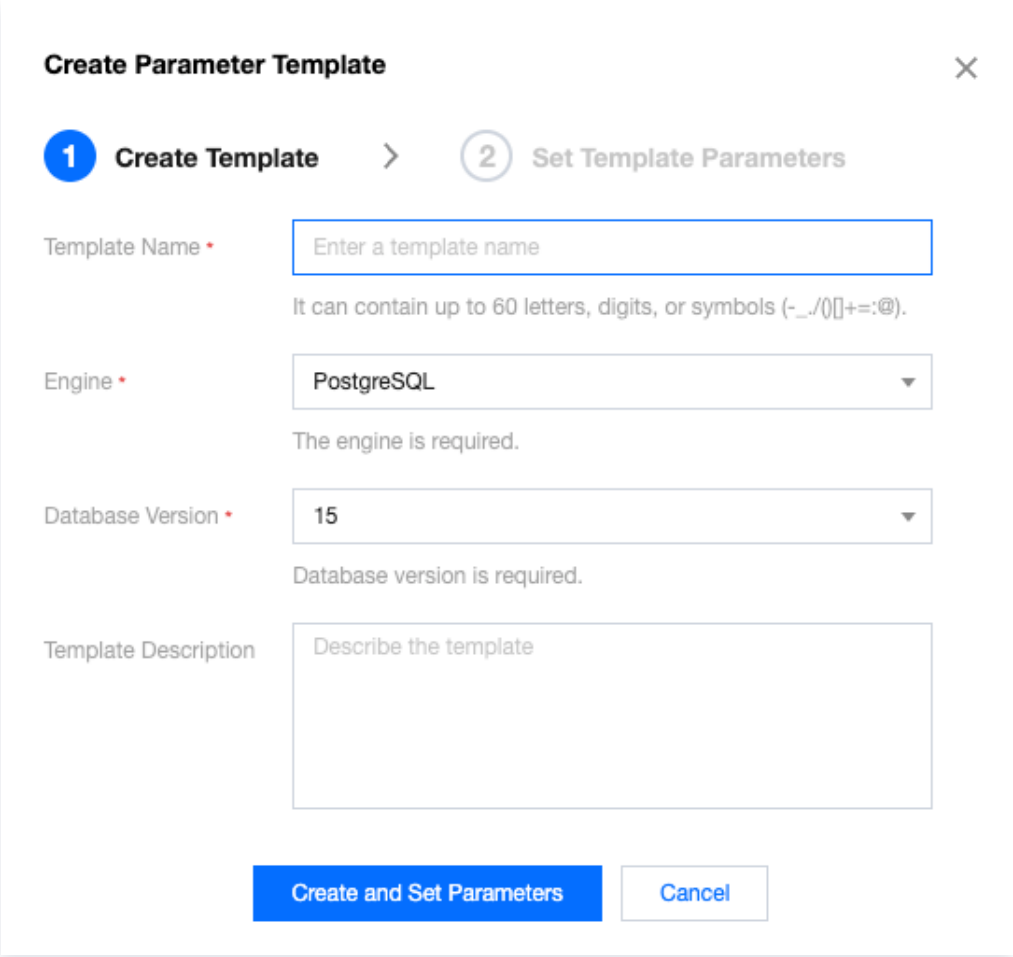
- If the parameters in the template are updated, the instance parameters are not updated unless they are manually re-applied to the instances.
- You can apply the parameter changes to single or multiple instances by importing a template.

Creating a Parameter Template

To utilize your own database parameter template, simply create a new template, modify the required parameters, and apply it to your database instances.

1. Log in to the TencentDB for PostgreSQL console, select **Parameter Templates** on the left sidebar, and click **Create Template**.
2. In the pop-up window, configure the following parameters and click **Create and Set Parameters**.

- Template Name: Enter a unique name for the parameter template.
- Engine: Select the appropriate database engine, such as PostgreSQL or SQL Server compatible.
- Database Version: Select the desired database version.
- Template Description: Enter a brief description of the parameter template.

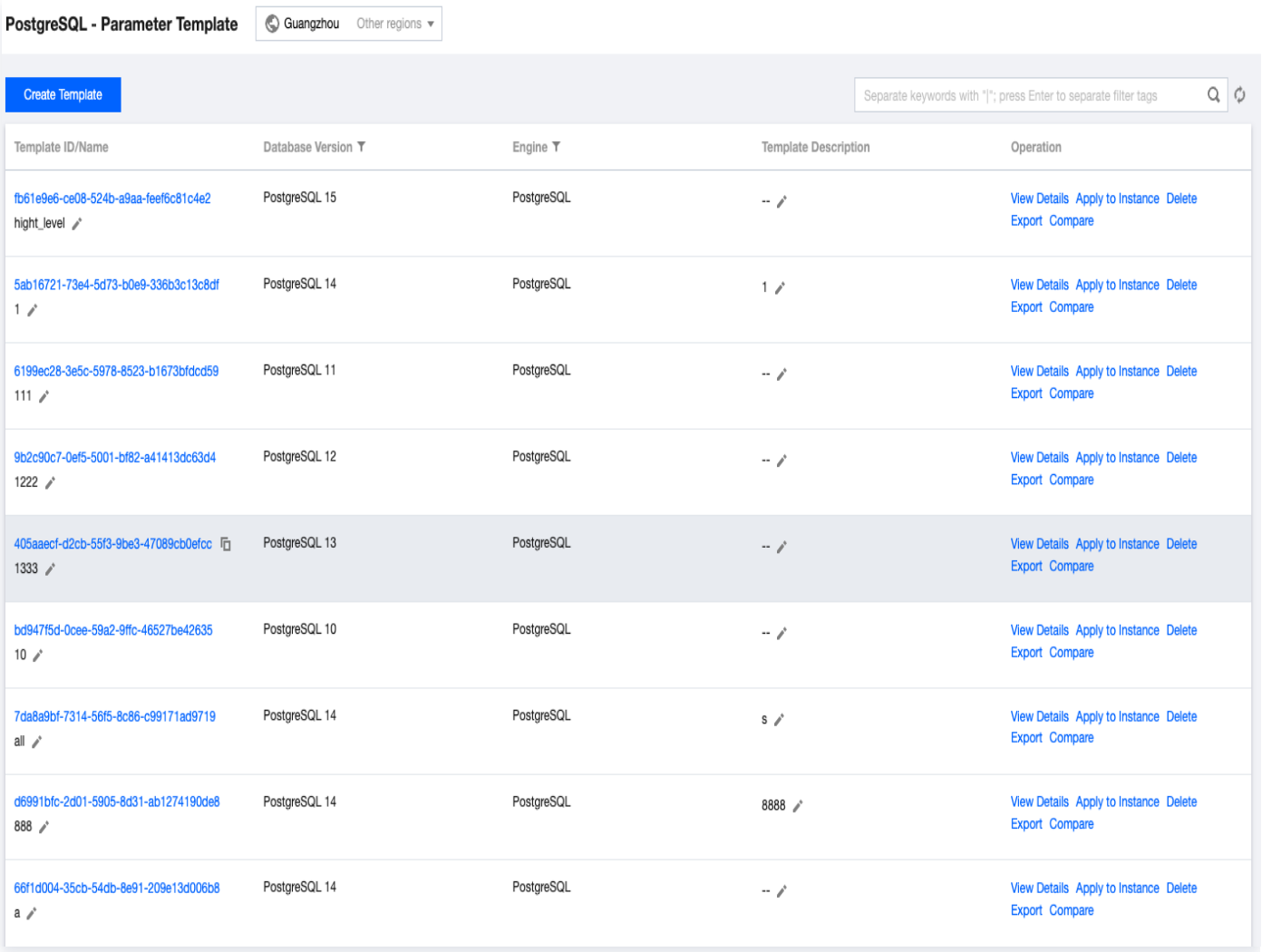


3. After the creation is completed, you can modify, import, and export parameters on the template details page.

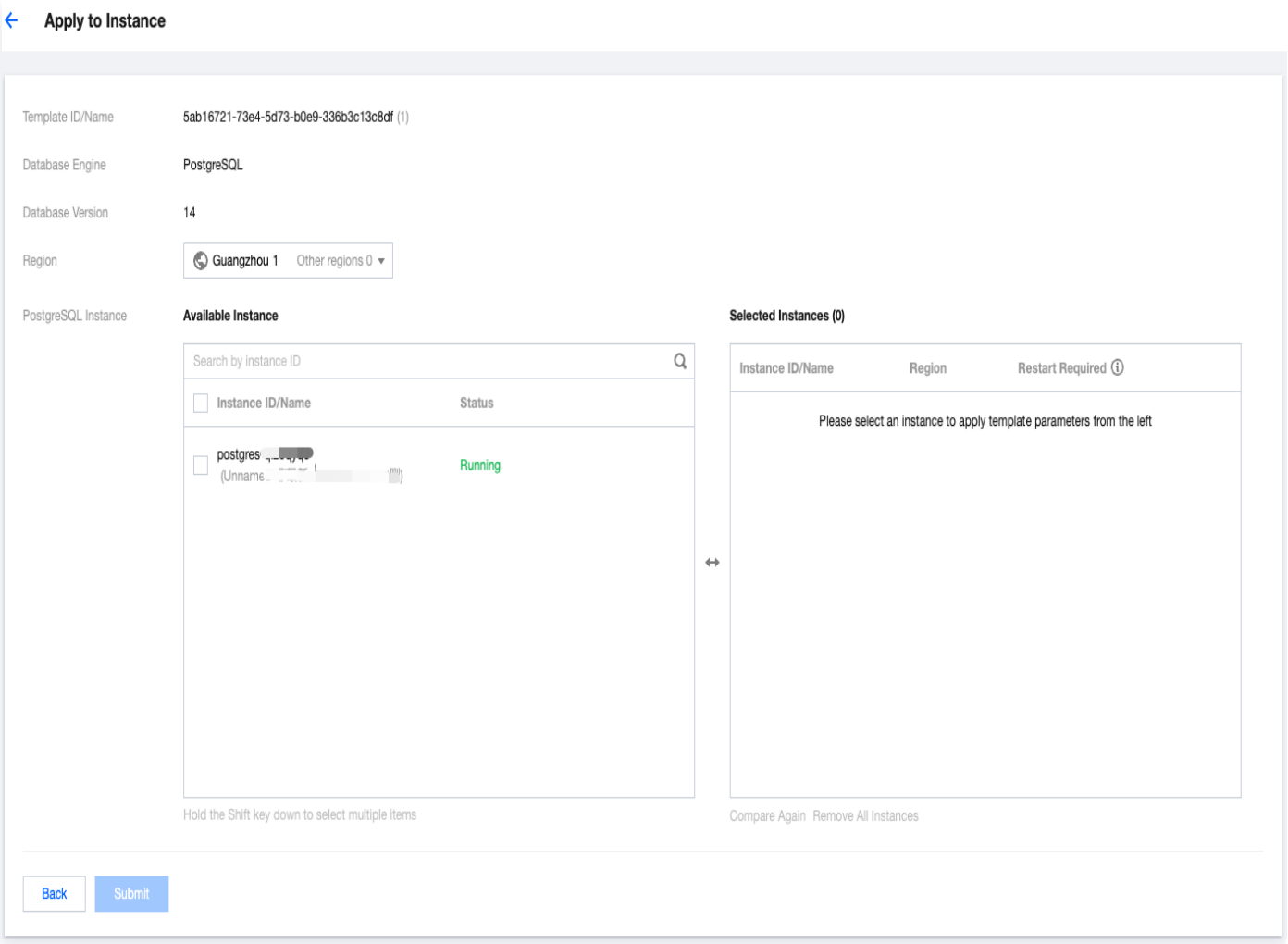
Applying a Parameter Template to Instances

> **Note**
> Before applying a parameter template to multiple instances, please make sure that the instances do support those parameters.

1. Log in to the TencentDB for PostgreSQL console and select **Parameter Template** on the left sidebar.
2. In the parameter template list, locate the desired template and click **Apply to Instance**.



3. On the displayed page, specify the execution mode and instances, make sure that all parameter values are correct, and click **Submit**.
   - PostgreSQL Instance: Select one or more instances that need to apply the parameter template in the specified region.
   - Parameter Comparison: View the changed parameter values of the selected instance.

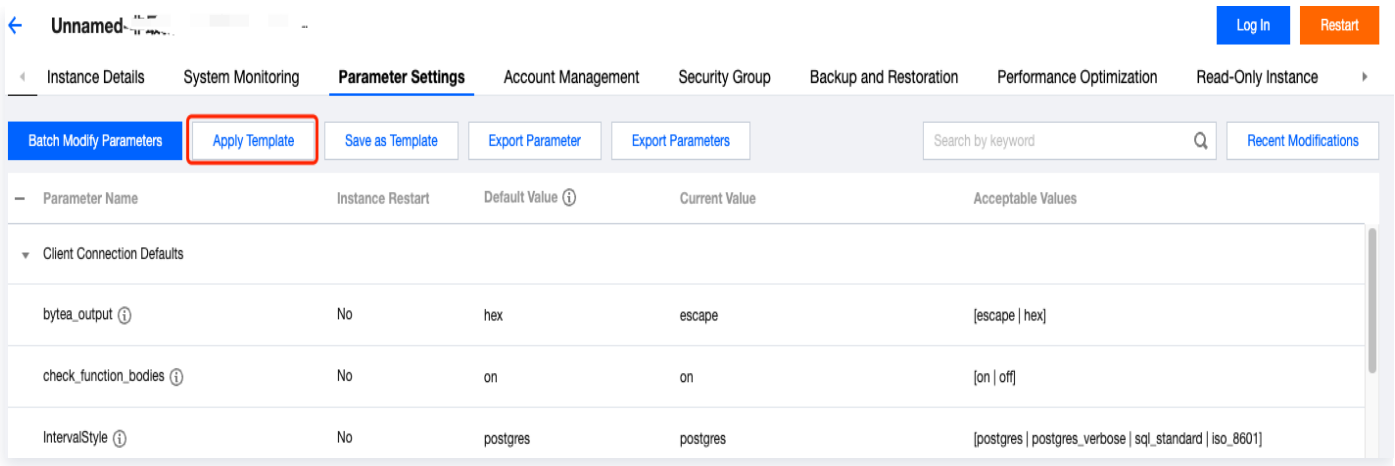


Copying a Parameter Template

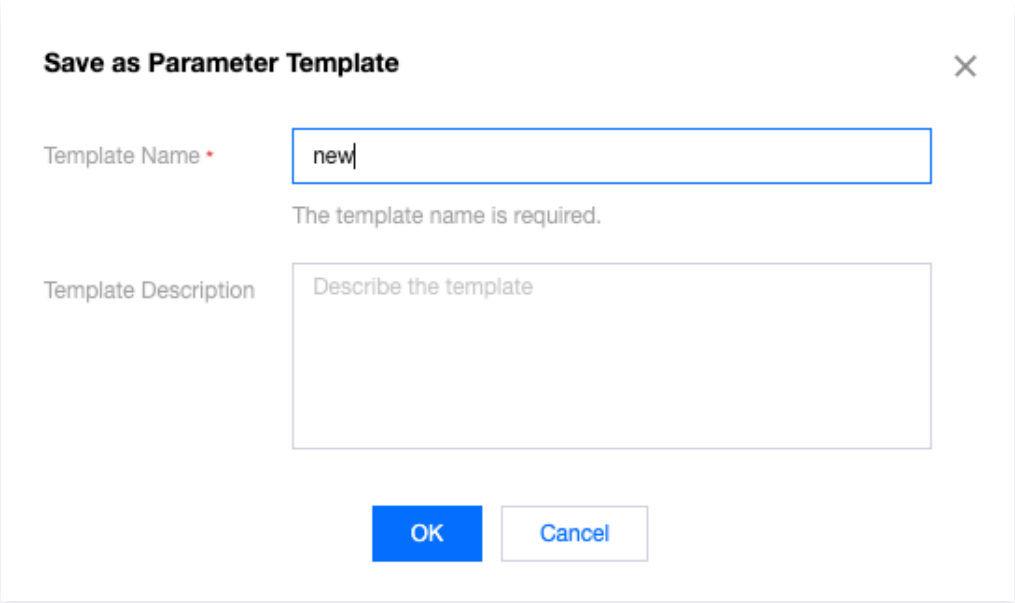
To include most of the custom parameters and values of an existing parameter template in a new template, you can copy the existing template.

Method 1. Copying an existing parameter template

1. Log in to the TencentDB for PostgreSQL console, select **Parameter Templates** on the left sidebar, and click the template ID or **View Details** in the **Operation** column to enter the template details page.
2. Click **Save as Template** on the template details page.



3. In the pop-up dialog box, specify the following configurations:
   - Template Name: Enter a unique name for the parameter template.
   - Template Description: Enter a brief description of the parameter template.



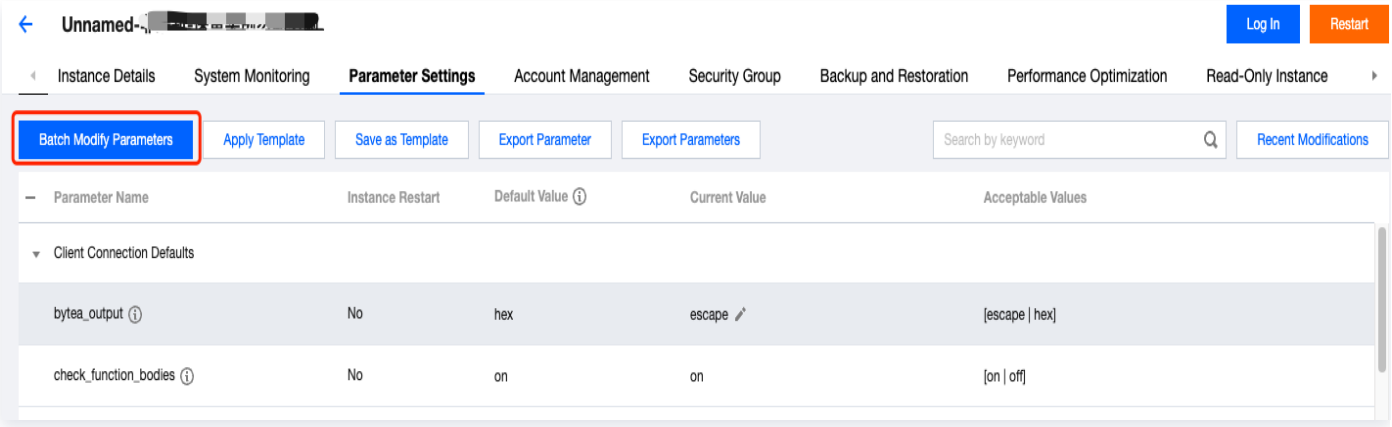
4. After verifying the information, click **OK** to save the current parameter template as a new template, completing the duplication process.

Method 2. Saving parameters of an instance as a parameter template

1. Log in to the TencentDB for PostgreSQL console, select **Instance List** from the left navigation, and click the instance ID to enter the management page.
2. On the **Parameter Settings** tab, click **Save as Template**.
3. In the pop-up dialog box, specify the following configurations:
   - Template Name: Enter a unique name for the parameter template.
   - Template Description: Enter a brief description of the parameter template.
4. After verifying the information, click **OK** to save the current parameter template as a new template, completing the duplication process.

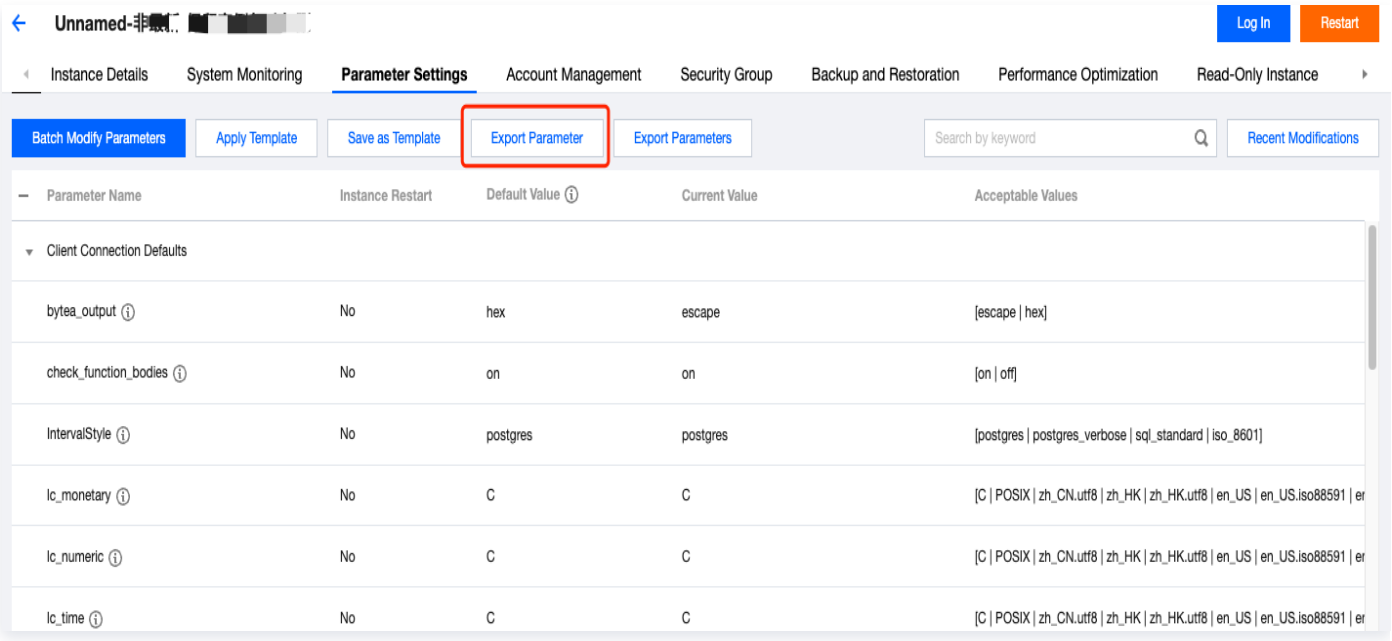
Modifying Parameter Values in a Parameter Template

1. Log in to the TencentDB for PostgreSQL console, select **Parameter Templates** on the left sidebar, click the template ID, and enter the template details page.
2. In the template details page, click **Batch Modify Parameters**, or click the "Modify Parameter Value" icon in the parameter current value column.



Importing a parameter template

1. Log in to the TencentDB for PostgreSQL console, select **Parameter Templates** on the left sidebar, click the template ID, and enter the template details page.
2. In the template details page, click **Import Parameters**.



Note

When selecting to import parameters, on the Import from Local File page, ensure that the selected parameter configuration file has the same format as the PostgreSQL database server configuration file or uses the exported parameter file template; otherwise, the import will be unsuccessful.

3. In the pop-up window, select the file and click **Import and Overwrite Original Parameters**.

Exporting a parameter template

Method 1

1. Log in to the TencentDB for PostgreSQL console and select **Parameter Template** on the left sidebar.
2. In the parameter template list, locate the desired template and click **Export** in the **Action** column.

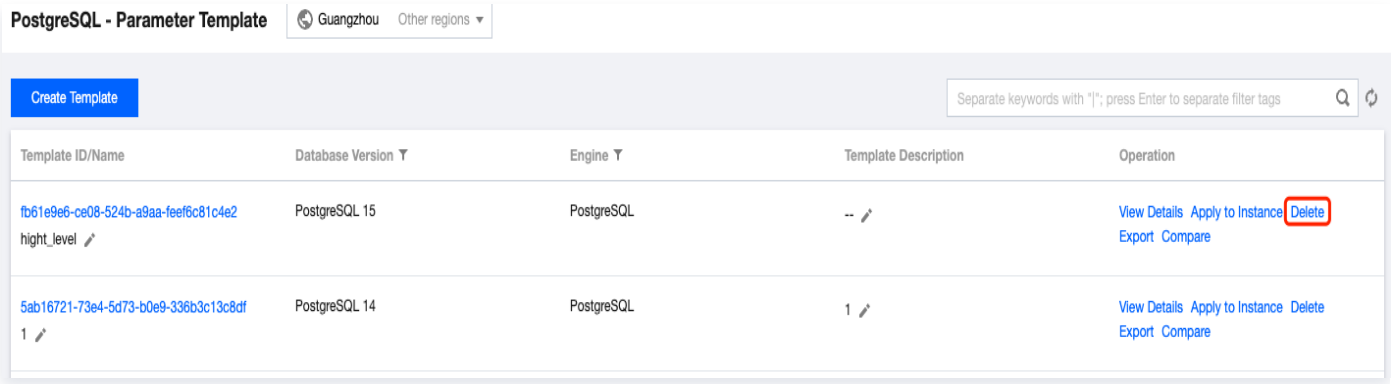
Method 2

1. Log in to the TencentDB for PostgreSQL console, select **Parameter Templates** on the left sidebar, click the template ID, and enter the template details page.
2. On the template details page, click **Export Parameters** at the top.

Deletes a parameter template.

If a parameter template is created redundantly or no longer needed, it can be easily deleted.

1. Log in to the TencentDB for PostgreSQL console and select **Parameter Template** on the left sidebar.
2. In the parameter template list, click **Delete** in the **Operation** column for the desired template.



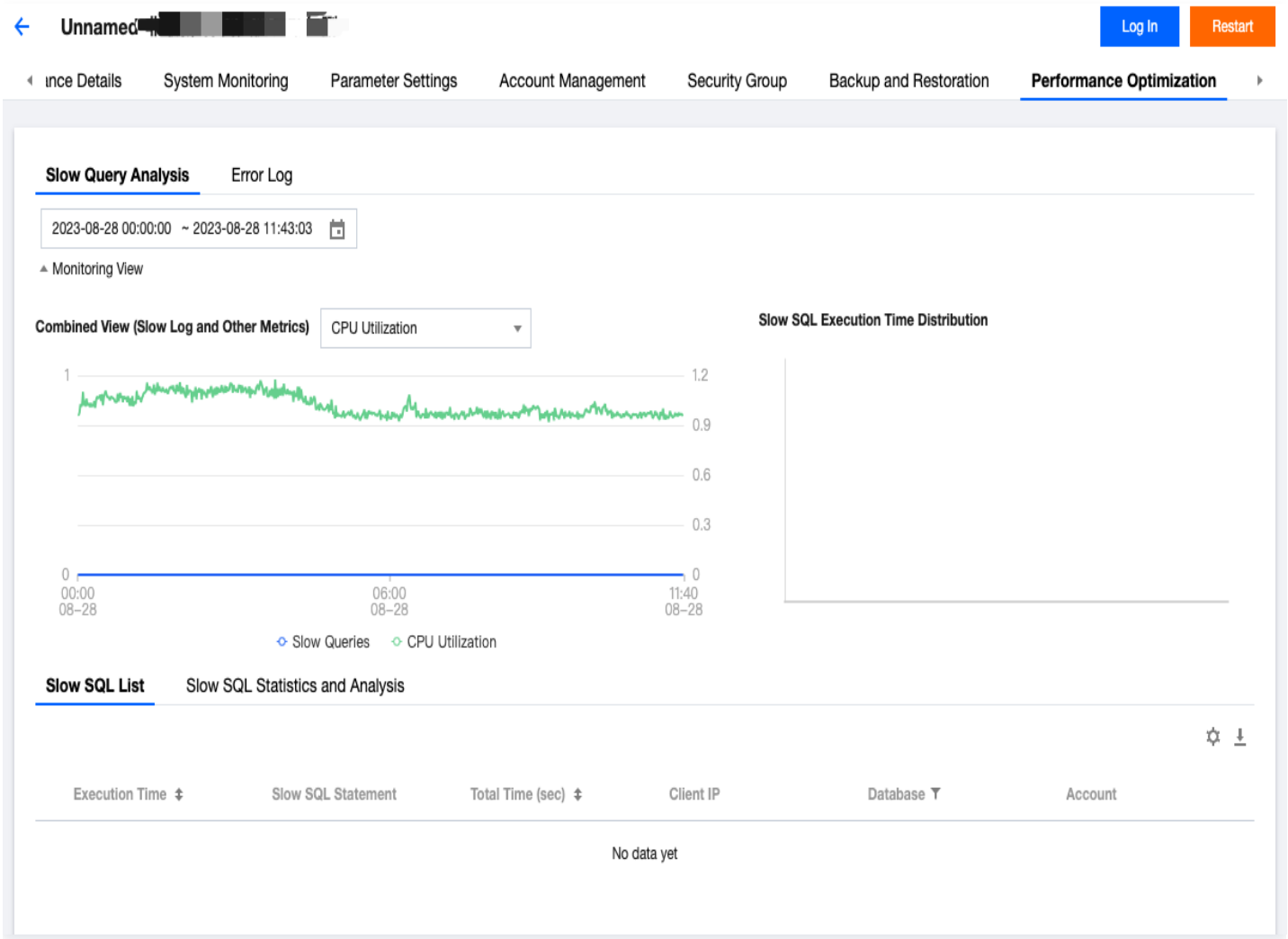
3. In the pop-up dialog box, click **OK**.

Log Management and Analysis
Slow query analysis

Last updated: 2023-09-10 09:03:23

Note

By default, a SQL query that takes more than one second is a slow query, and the corresponding statement is a slow query statement. The process where a database administrator (DBA) analyzes slow query statements and finds out the reasons why slow queries occur is known as slow query analysis.
The TencentDB for PostgreSQL console provides slow query analysis capabilities under the **Performance Optimization** module on the instance management page, as shown below:



Monitoring View

The console features two charts as monitoring views, allowing you to intuitively and conveniently view the slow SQL-related information of the database.
Slow Query and Other Monitoring Combined View: This view supports visualizing slow query metrics

and comparing them with other metrics. Other supported metrics include: CPU utilization, QPS, request count, read request count, write request count, other request count, buffer cache hit rate, and average execution latency.
Slow SQL Duration Distribution: This view enables you to examine the primary distribution of slow queries within different time intervals by analyzing slow SQL statements across various time periods.

Slow SQL List

The slow SQL list shows slow query statements of the database in real time. The list is arranged in descending order by time, that is, the latest slow query statement is automatically displayed in the first row.
The slow SQL list has the following fields: the execution time, the slow SQL statement, the total time, the client IP, the database name, and the account executing the statement.

> **Note**
> - By default, the slow SQL list displays slow SQL data over the past seven days. The slow SQL data is stored in a log, and the oldest data is automatically deleted from the log to ensure that the log only stores data within the past seven days and the log size does not exceed 50 GiB.
> - If a single slow SQL statement exceeds 20KB, it cannot be viewed from the console. Please submit a ticket and contact Tencent Cloud for assistance.

Slow SQL Statistics and Analysis

Slow SQL statistical analysis is based on the aggregation of similar SQL statements after abstracting system parameters from all slow SQL statements within a specified time range. This analysis provides slow SQL information, which includes various field information.

- **Last Execution Time**: Within the statistical scope, this is the time when the abstract statement last appeared. Since some statements are expected to have a longer execution time, we uniformly record the begin_time of the statement execution.
- **Abstracted SQL Statement:** This refers to the statement after removing constants from the slow SQL. Abstracted statements allow for the aggregation and summary of similar statements, making it easier for you to analyze them effectively.
- **Database**: The database that the statement has accessed.
- **Account**: This indicates which account was used to execute the statement.
- **Client IP Address**: The client locations where the statement has been executed.
- **First Execution Time**: The time when the slow SQL statement first appeared within the statistical range (after abstraction and aggregation, there may be multiple records).
- **Total Execution Time**: The total duration of slow query statements within the statistical scope.
- **Average Execution Time**: The average time is calculated by dividing the total time consumed by the slow query statement by the total number of executions of the slow query statement.
- **Minimum Execution Time**: The minimum execution time among all instances of this abstract

statement as a slow query statement; used to help determine whether the statement is an occasional event.

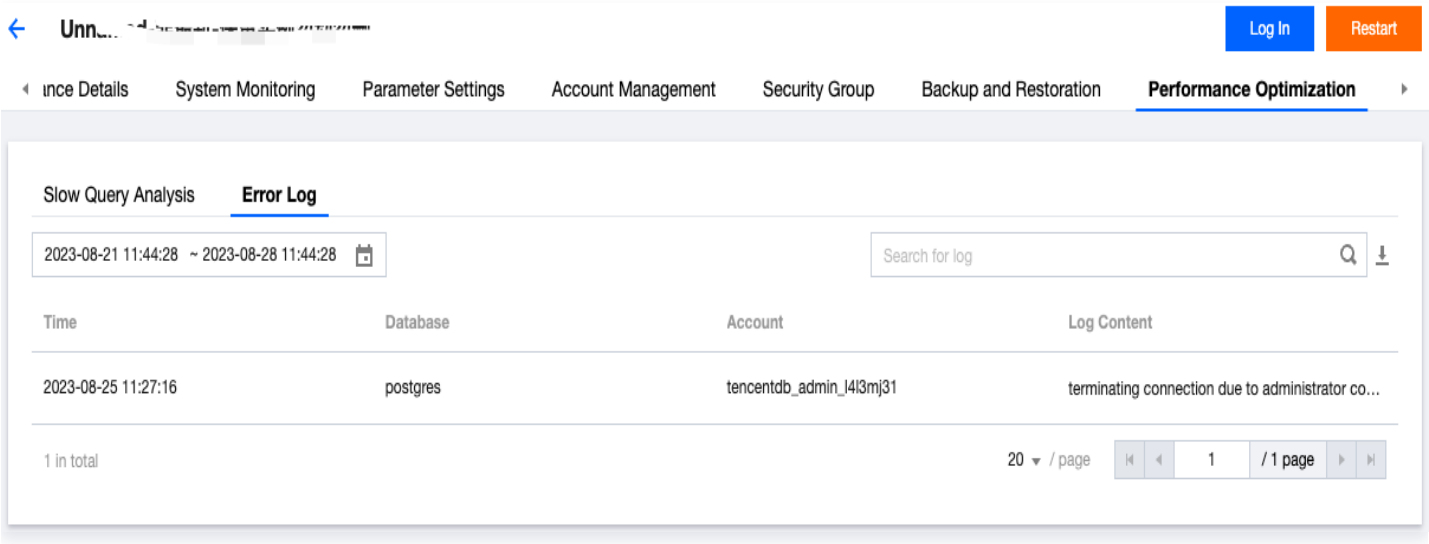
- **Maximum Execution Time**: The maximum execution time among all instances of this abstract statement as a slow query statement; used to help determine whether the statement is an occasional event.
- **Total Time Ratio**: Within the statistical scope, the proportion of time taken by a slow query statement compared to the total time taken by all slow query statements.

Error Logs

Last updated: 2023-09-10 09:03:31

Note

Error logs refer to logs generated due to operation, SQL, and system errors during database running. Error logs are usually used by developers to find out the causes of errors in business systems or databases.
In TencentDB for PostgreSQL, the error log viewing capability is provided under the **Performance Optimization** module on the Instance Management page, as shown below:



Default Settings of Error Log

- Error log feature: enabled by default.
- Error log level: `log_min_error_statement=ERROR` .
- Analyzed data output delay: 1-5 minutes.
- Log retention period: 7 days (up to the last 10,000 records).

Managing Operational Logs

Last updated: 2023-09-10 09:03:41

This document describes how to manage the retention period of pg_log.

Introduction to PG_LOG

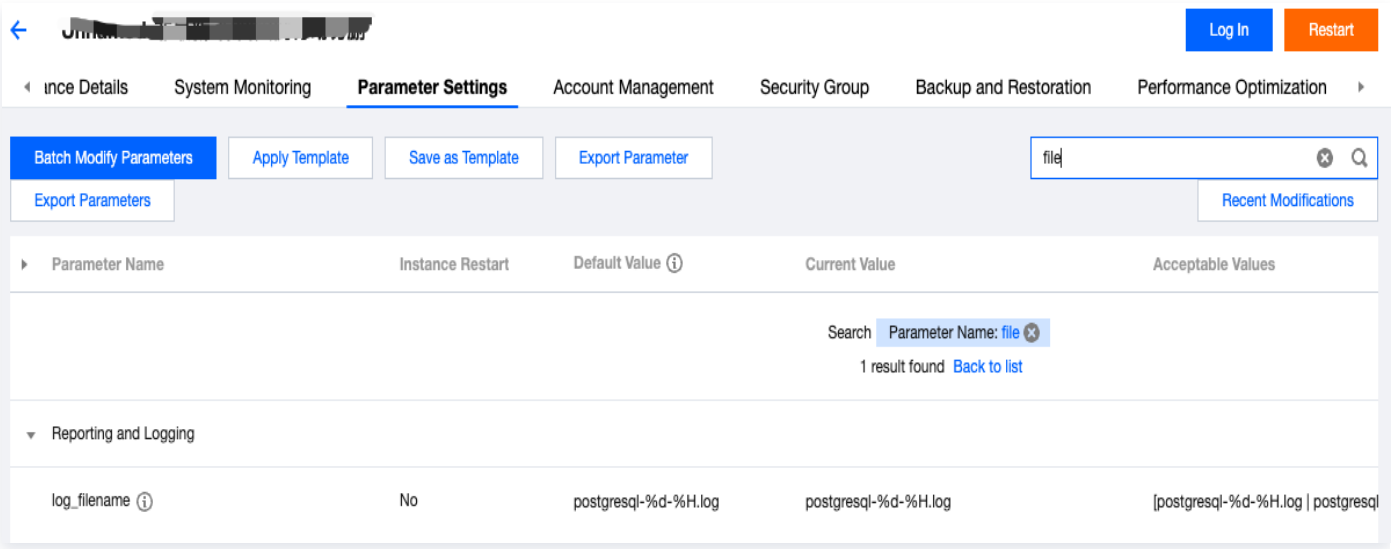
pg_log typically records the status information of a database, such as error messages, slow query SQL statements, and database startup and shutdown information. The log is automatically split by size and time, and the default retention period for pg_log in TencentDB for PostgreSQL is 30 days. pg_log occupies the storage space of the database instance, and you can modify the retention period based on your actual needs.

> **Note:**
> The slow logs and error logs of the database instance are retained for seven days by default. Modifying the retention period of pg_log does not affect the retention period of slow logs and error logs.

Change PG_LOG Retention Period

You can modify the retention period of pg_log based on your actual needs. The system currently supports two options: 7 days and 30 days. The specific steps are as follows:

1. Log in to the TencentDB for PostgreSQL console.
2. In the instance list, locate the instance you want to modify, click **Actions** > **Manage**, and enter the instance details.
3. On the Instance Details page, locate **Parameter Settings**. In the search box at the top right, search for the parameter log_filename that needs to be set and update it.



The description of the log_filename parameter is as follows:

Value	Note
postgresql_%a_%H.log	By selecting this option, logs will be retained for seven days.
postgresql_%d_%H.log	By selecting this value, logs will be retained for 30 days, which is the system's default retention period.

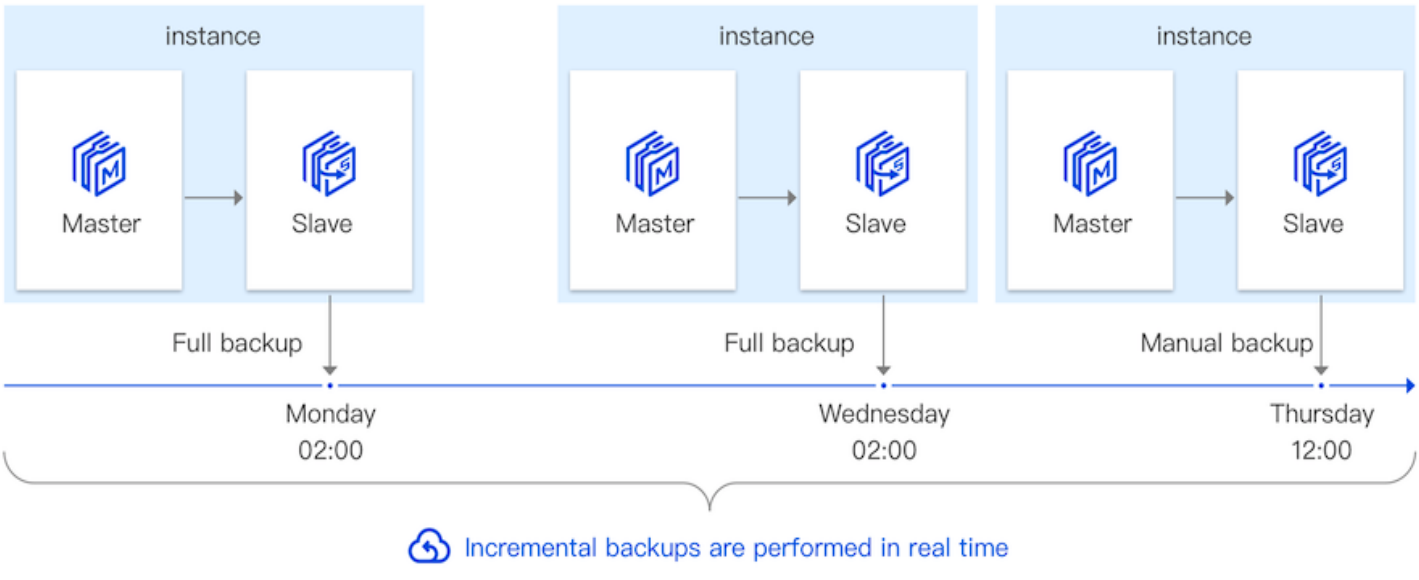
Backup and Restoration
Backup Principles and Solutions

Last updated: 2023-09-10 09:04:12

TencentDB for PostgreSQL supports **automatic full backups**, **manual full backups**, and **log backups**. You can schedule regular database backups through automatic backup settings. In case of database failure or accidental deletion, you can restore the database using the stored backup files. TencentDB for PostgreSQL stores database backups in a compressed format, with the compressed backup files being approximately 30% of the original size (the actual compression ratio depends on the instance's stored data, with higher redundancy resulting in a larger compression ratio).

Backup Principles

For dual-server high-availability (primary and secondary) architecture, when a backup task is triggered, the system retrieves data from the secondary instance and compresses it before uploading it to the object storage service. Backup space does not occupy the instance's disk space. In case of database failure or accidental deletion, you can restore the database using the stored backup files. You can use the cloning method for data recovery, as detailed in Clone Instance, or download the backup files for restoration, as explained in Restoring PostgreSQL Data on a Cloud Server. The principle is illustrated in the following diagram:



Backup Solutions

Item	Backup Type	Details

Data Backup	Automatic Full Backup	• The system will trigger a full data backup within the specified time based on your automatic backup settings. The backup method used is physical backup, which offers faster backup speed and higher recovery efficiency. • The generated full automatic backup data is retained based on the data backup retention period you set. This backup data will not be deleted when the instance is terminated and will be automatically deleted after expiration. This meets your requirement to retain backup data for an extended period to prevent severe impact due to accidental instance deletion. As full backups occupy backup storage space, you can delete them promptly if needed. • Automatic backup data within one week cannot be deleted, while automatic backup data older than one week can be flexibly removed based on your needs. Once the backup data is deleted, it cannot be recovered, so please proceed with caution.
	Manual Full Backup	• You can initiate manual backups in the console based on your application requirements. Once a manual backup task is initiated, the system will perform a full instance backup using the physical backup method within 1 minute. • The expiration time for manual backups is one week after initiation. Since manual backups occupy backup storage space, you can delete them promptly if needed.
	Incremental Backup	• Incremental backup refers to WAL log backup, which is automatically enabled by default and cannot be disabled. Incremental backups are retained based on the data backup retention period you set. The system performs real-time backups based on the WAL logs generated by the database. As incremental backups occupy backup storage space, you can delete them promptly if needed. • Incremental backup data within a week cannot be deleted, while incremental backup data older than a week can be deleted as needed. Please exercise caution when deleting incremental backup data, as it will make point-in-time recovery impossible.
Download Backup File	Download Full Backup	Supports local browser download and download by address.
	Download Incremental	Supports local browser download and download by address.

	Backup	
Transferring historical backups via Cloud Functions	Transfer via Serverless	You can use Serverless Cloud Function (SCF) to transfer historical backup data. For more information, refer to Transferring PostgreSQL Historical Backups with SCF.

Backup Fees

TencentDB for PostgreSQL stores backup files as compressed packages, with the compressed backup files being approximately 30% of the original size. Backup space for a single instance is granted based on the purchased capacity, and most instances do not require payment. For specific billing rules, please refer to Backup Space Billing.

Data Backup

Last updated: 2023-09-10 09:04:21

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

Scenario

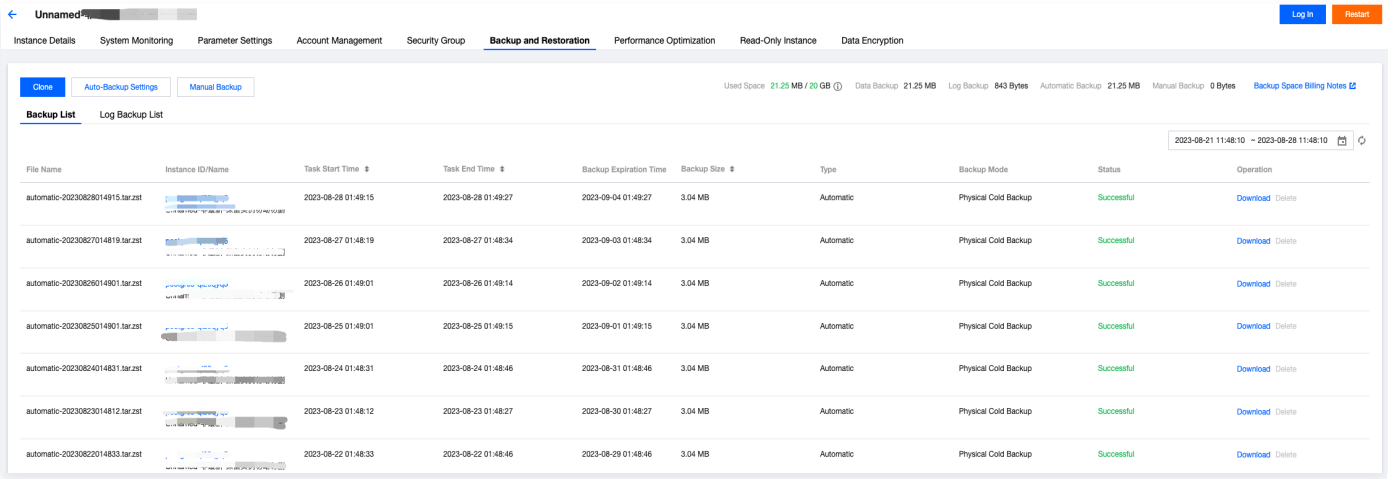
TencentDB for PostgreSQL High-Availability Edition currently supports physical backup methods:

- **Full Backup**: Back up once daily at 01:00.
- **Incremental Backup**: Back up every 15 minutes when an xlog is generated; or back up when an xlog is generated and has accumulated up to 60 xlogs.
- **Data File Retention Period**: Supports setting backup data retention for 7 to 1,830 days.
- **Manual Backup**: Users can initiate backup tasks themselves, which will be launched within 1 minute after clicking.

Instructions

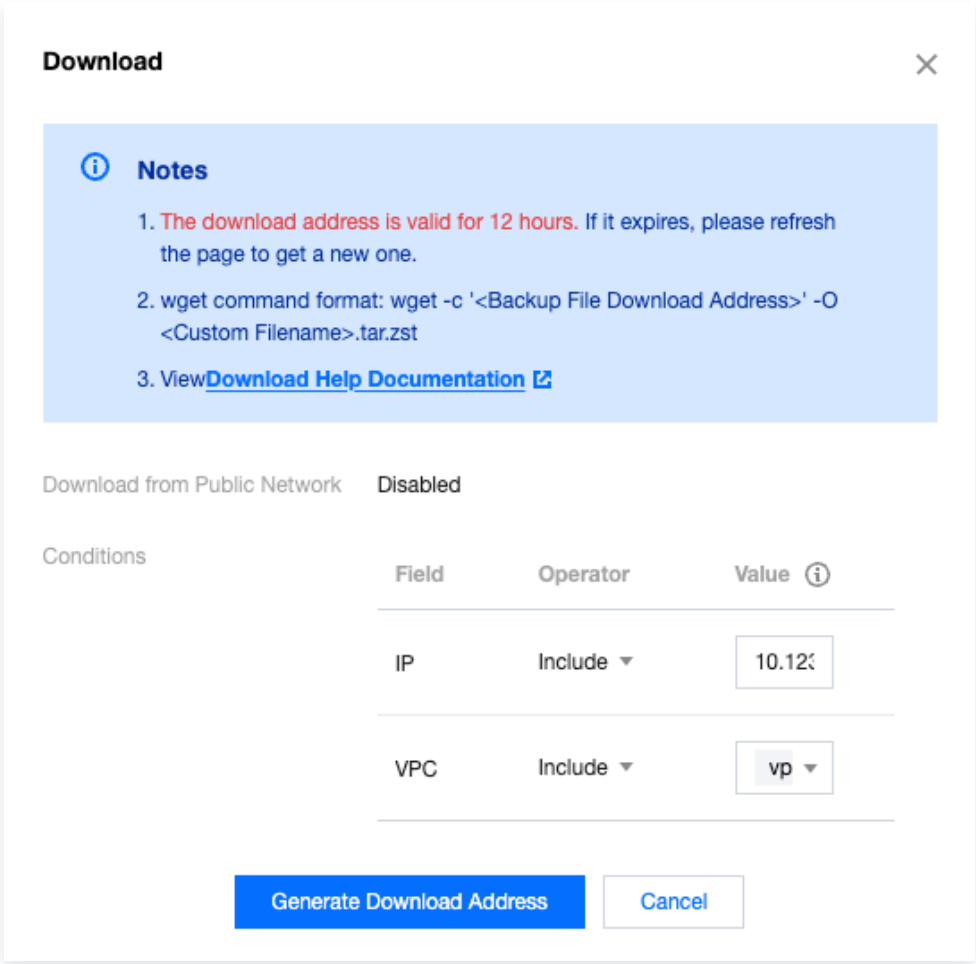
1. Log in to the TencentDB for PostgreSQL Console, click the instance ID, and enter the instance management page.
2. On the instance management page, select the **Backup and Restoration** tab, choose either **Backup List** or **Log Backup List**, locate the backup you want to download, and click **Download** in the **Operation** column.

> **Note**
> **Backup List** contains full backup data; **Log Backup List** contains incremental backup data.



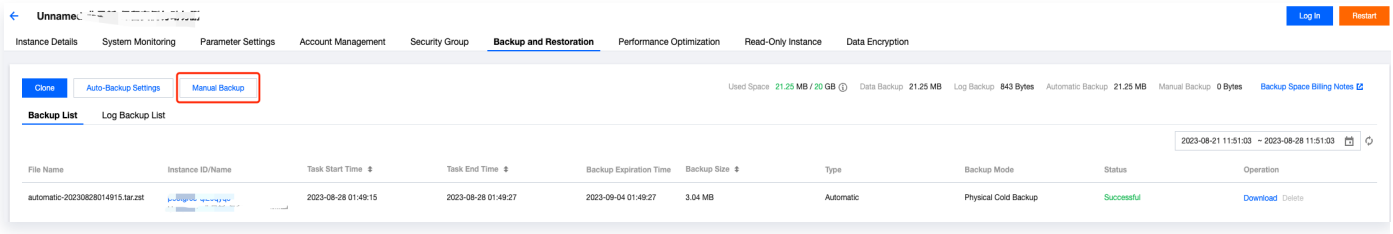
3. In the pop-up dialog box, two download methods are provided: Private VPC Network Address and Local Download.

> **Note**
> To ensure data security, the address is valid for 12 hours. If it expires, refresh the page to get a new one. Please access the VPC network address within the VPC network.

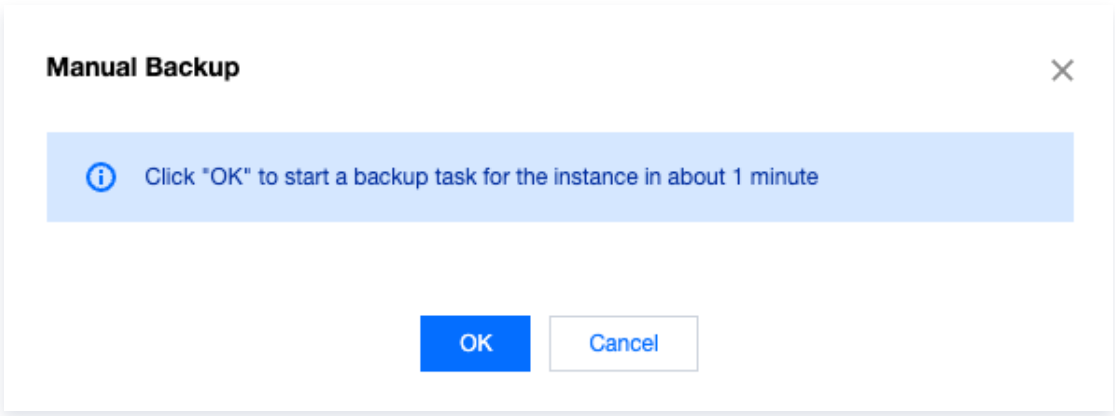


4. Users can initiate backup tasks as needed. Click the **Manual Backup** button on the **Backup and Restoration** page to initiate a manual backup.

> **Note:**
> - During the daily automatic backup task, manual backups cannot be initiated for the instance.
> - Manual backups can be deleted from the backup list to free up backup space and avoid wasting space and resources. If not manually deleted, they will be retained until the database instance is taken offline.



5. In the **Manual Backup** dialog box, click **OK**.



> **Note:**
> The time it takes to create a manual backup is related to the actual capacity of the instance. If the capacity is large, the manual backup will take longer to complete.

FAQs

Can backups beyond the retention period still be downloaded or restored?

Expired automatic backup sets will be automatically deleted and cannot be downloaded for restoration. You can manually back up instance data through the console, and manual backups will be preserved indefinitely.

Can backups be manually deleted?

Automatic backups within 7 days cannot be deleted, while those older than 7 days can be deleted as needed. The system will also automatically delete them based on the automatic backup retention period. Manual backups can be deleted manually from the backup list, and if not deleted, they will be retained indefinitely.

Can data and log backups be disabled?

You cannot disable backups. However, you can reduce backup space usage by decreasing the backup frequency and deleting unnecessary manual backup data.

How to reduce backup space costs?

- Delete unnecessary manual backup
- Reduce the automatic backup frequency for non-core business data (at least twice a week).

Downloading Backups

Last updated: 2023-09-10 09:04:28

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

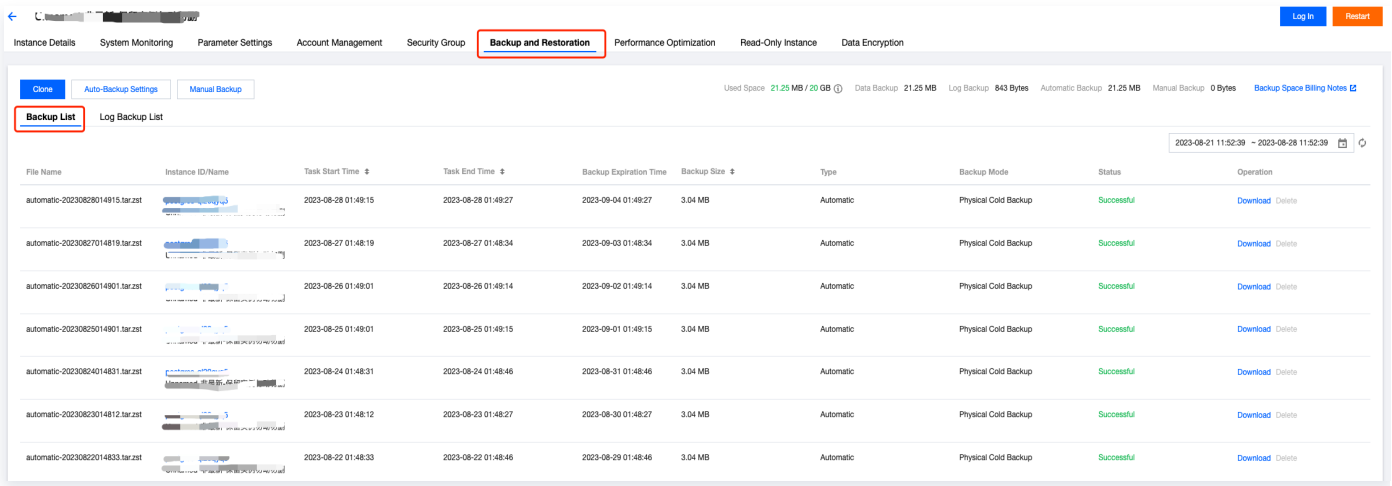
Supports and Limits

- Both private address and local download address are valid for 12 hours. Refresh the page to get new ones upon expiration.
- The URL must be enclosed with quotation marks when the `wget` command is used to download.

Instructions

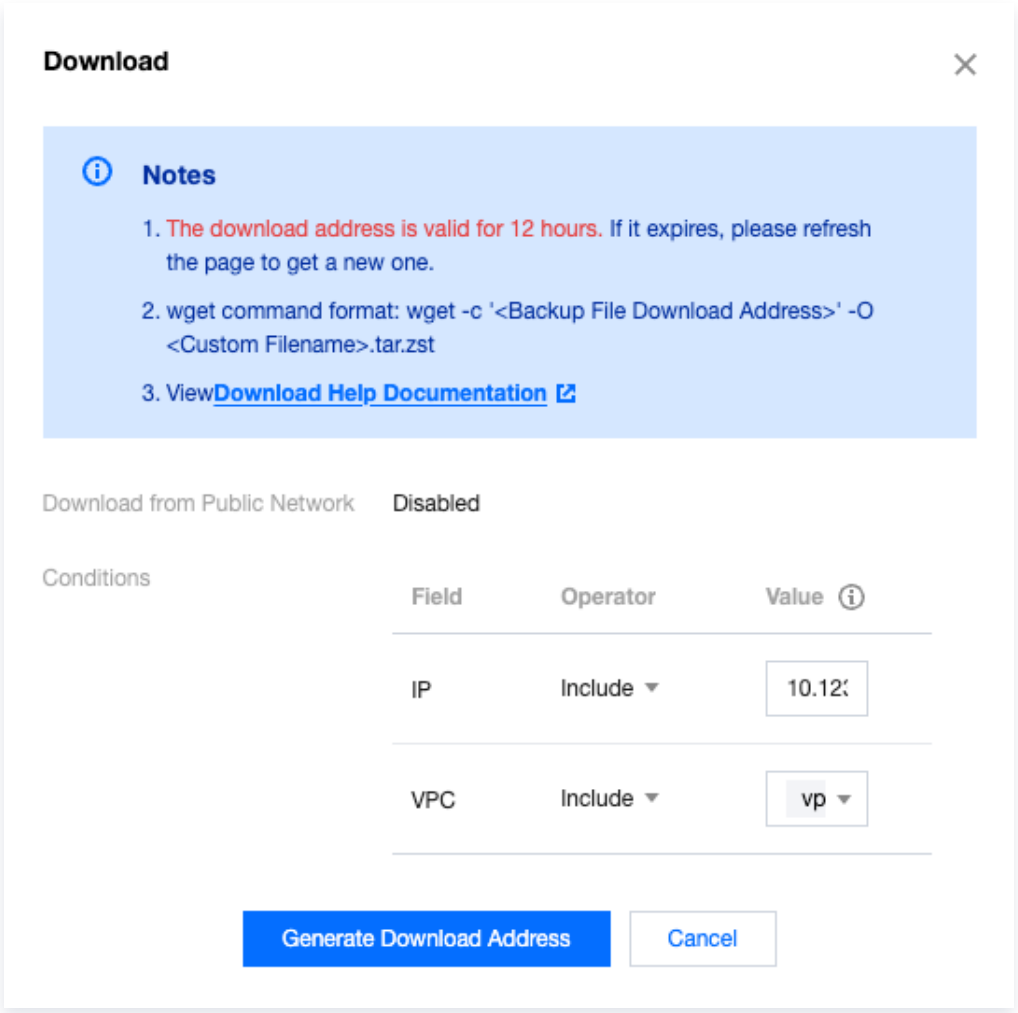
Data backup download

1. Log in to the TencentDB for PostgreSQL console, select the region, click the instance ID, and enter the instance management page.
2. On the instance management page, select the **Backup and Restoration** tab and click **Backup List**.



3. In the backup list, locate the desired backup file and click **Download** in the **Operation** column.
4. In the download window, both VPC private address and public address are provided for download.

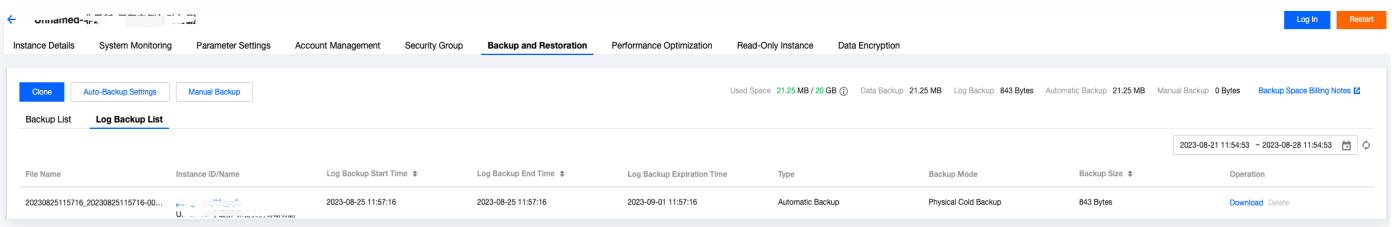
> **Note**
> It is recommended to copy the download address and log in to the CVM (Linux system) under the VPC where the cloud database is located. Utilize the `wget` command for high-speed intranet downloads, which is more efficient.



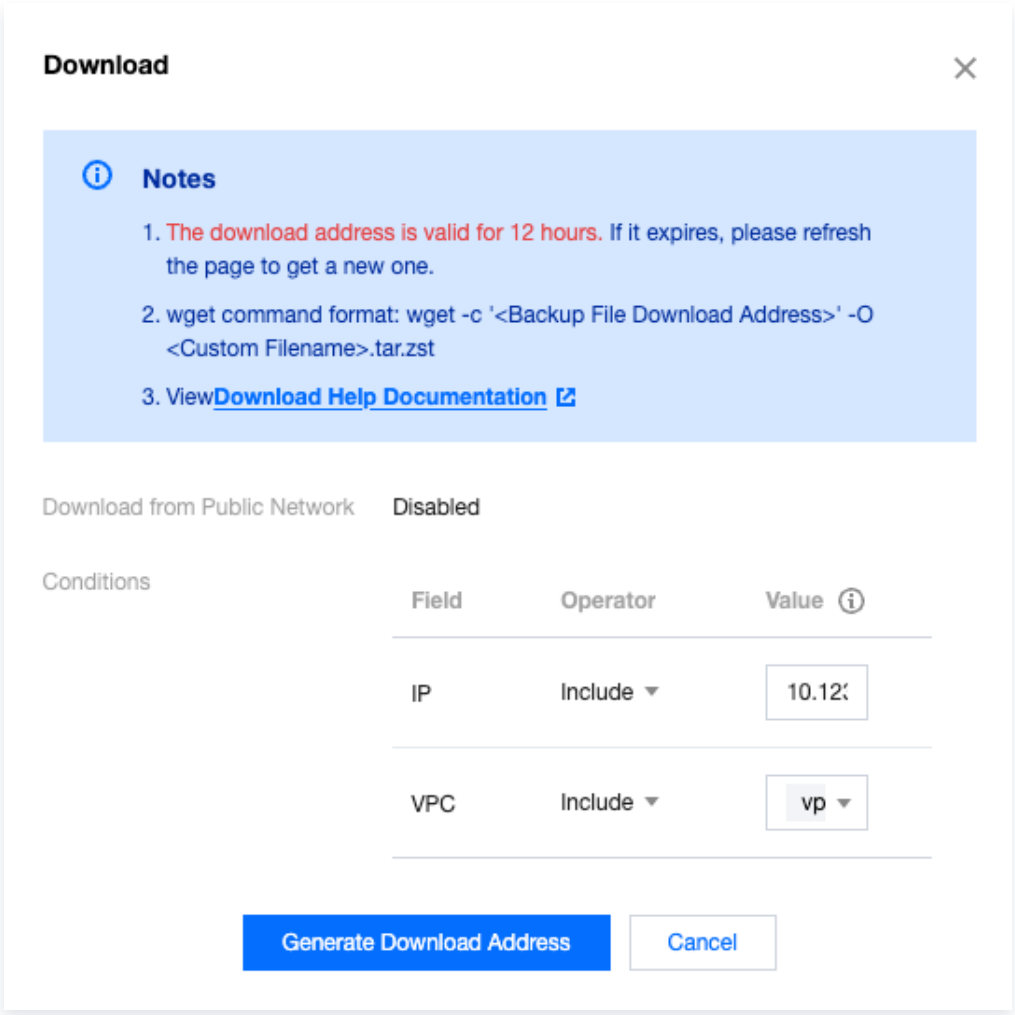
5. You can also click **Database Backup** on the left sidebar to enter the **Backup List** page, and click **Data Backup List** to download the data backup.

Log backup download

1. Log in to the TencentDB for PostgreSQL console, select the region, click the instance ID, and enter the instance management page.
2. On the instance management page, select the **Backup and Restoration** tab and click **Log Backup List**.



3. In the Log Backup List, locate the desired backup file and click **Download** in the **Operation** column.
4. In the download window, both VPC private address and public address are provided for download.



5. You can also click **Database Backup** on the left sidebar to enter the **Backup List** page, and click **Log Backup List** to download the data backup.

Cloning Instance

Last updated: 2023-09-10 09:04:35

If you need to perform data recovery, you can use the Clone PostgreSQL Instance feature to quickly restore instance data from a backup to a newly purchased PostgreSQL instance. This document guides you on how to perform cloning operations through the TencentDB for PostgreSQL console.

Scenario

TencentDB for PostgreSQL offers a Clone Instance feature, which supports restoring an instance to any point within the log backup retention period through cloning, and also supports restoring to a specified physical backup set. Cloning first creates a new instance based on the user-selected point in time and backup data. After validation, users can migrate the data back to the source instance using the DTS (Data Transmission Service), or directly use the cloned new instance.

Clone mode

- Clone an instance and restore the clone to any point in time within the log backup retention period you set.
- Clone an instance and restore the clone from a specific physical backup set within the data backup retention period you set.

Billing of the clone

- You can select a billing mode for the clone during the clone process in the same way as during instance purchase.
- The clone will not be billed until the clone process is completed.

Preparations

- The source instance must be in **Running** status.
- If you choose to clone by **Backup Set**, the source instance must have completed a physical backup. You can check the backup status in the Console Backup List.
- The balance of your account must be positive.

Supports and Limits

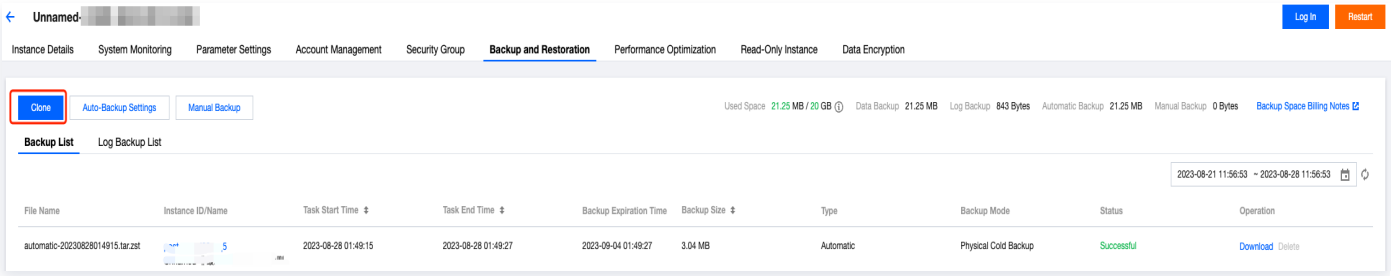
- The hard disk space of the clone must be larger than the amount of the data to be cloned from, or else the clone task may fail.
- The database version of the clone must be the same as that of the original instance.
- For instances with a used capacity exceeding 6TB, please submit a ticket for data recovery.

Instructions

1. Log in to the TencentDB for PostgreSQL console. In the instance list, click an instance ID or

Manage in the **Operation** column to enter the instance management page.

2. On the instance management page, select **Backup and Restoration** > **Backup List** tab, click **Clone** in the upper left corner, or click **Clone** in the **Operation** column of the backup you want to clone.



3. Navigate to the purchase page, select the clone mode and corresponding configurations for the cloned instance, and click **Buy Now** after confirming the details.

- **By time point**: You can restore an instance to any time point within the past seven days.
- **By backup set**: you can restore data from a backup set to a new instance. The available backup sets depend on the data backup retention period.

> **Note**
> You can view the backup retention period in the Console backup list.

4. After successful purchase, you can view the details of the clone on the instance list page.

FAQs

Will the access to the source instance be affected during the clone process?

The original backup set and log files uploaded to COS are used for restoration, which will not affect the access to the original instance.

Automatic Backup Settings

Last updated: 2023-09-10 09:04:43

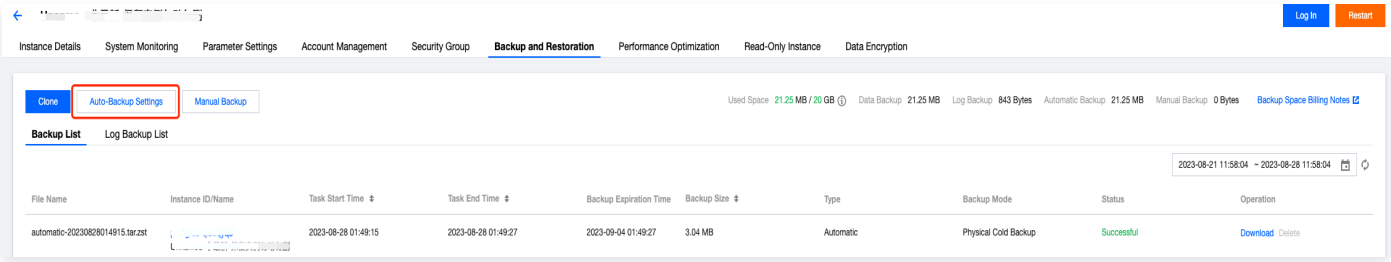
This document describes how to modify the automatic backup settings in the TencentDB for PostgreSQL console. TencentDB for PostgreSQL will automatically back up data based on the default or configured backup settings.

Supports and Limits

- Try to back up your data during off-peak hours.
- If the data volume is high, backup may take a long time. Please wait patiently.
- Backup files have a retention period. Please download the required backup files to your local storage in a timely manner, or use Cloud Functions to transfer PostgreSQL historical backups.
- The backup mode is physical backup, while logical backup is not supported currently.

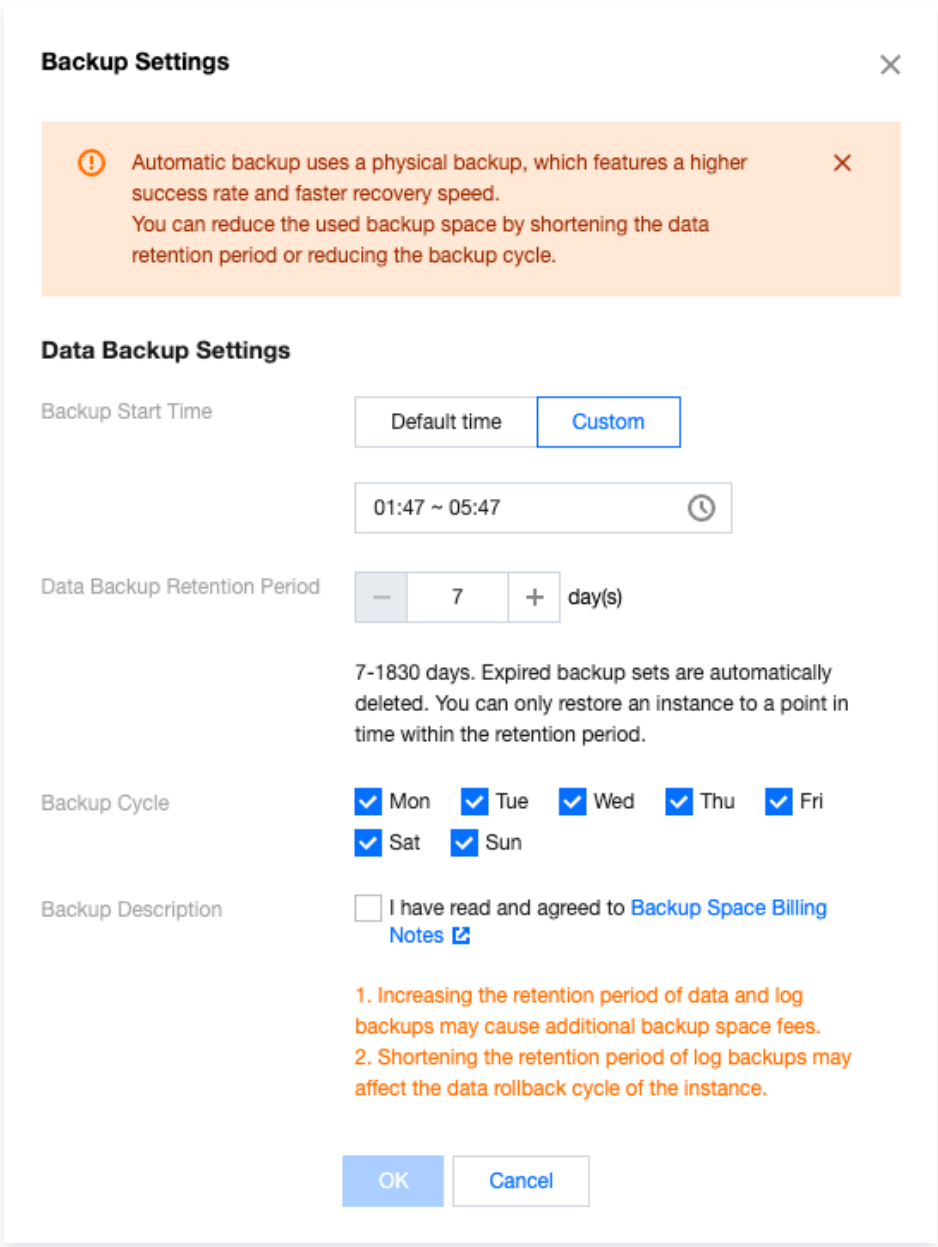
Instructions

1. Log in to the TencentDB for PostgreSQL console, select the region, click the instance ID, and enter the instance management page.
2. On the instance management page, select the **Backup and Restoration** tab and click **Auto-backup Settings**.

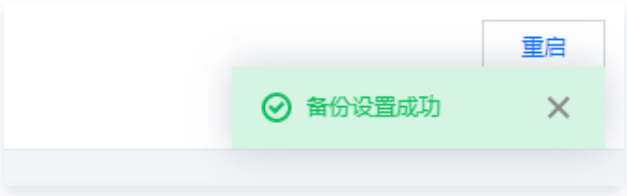


3. In the pop-up Backup Settings window, complete the data backup configuration and click **Confirm**.

- **Backup Start Time**: You can choose the default time (backups will be initiated during idle periods throughout the day) or customize the start time. Backups will be initiated within this time range. If a backup cannot be initiated due to unforeseen circumstances, it will not be restarted, and the next backup will be initiated during the next backup start time window.
- **Data Backup Retention Period**: You can set it between 7 and 1,830 days. Expired backup sets are automatically deleted. You can only restore an instance to a point in time within the retention period.
- **Backup Cycle**: Supports continuous or interval selection from Monday to Sunday.



4. When "Configured backup settings successfully" is displayed in the top-right corner, the automatic backup settings are completed.



Backup Configuration Explanation

Category	Note
Backup start time	• The default time is the backup initiation time automatically assigned by the system. • You can customize the backup start time, and it is recommended to set

	it during off-peak hours. The backup initiation time only indicates the start of the backup process, not the end time. For example, if you choose to enable backups between 02:00 and 06:00, the system will initiate a backup at some point within that time range, depending on the backend backup policy and the status of the backup system.
Data backup retention time	Backup files can be retained for 7 to 1,830 days, with a default retention period of 7 days. Expired backup sets are automatically deleted.
Backup cycle	By default, backups are scheduled for all seven days of the week (Monday to Sunday). You can customize the backup time, but for the sake of data security, please ensure that backups are performed at least twice a week.

Restoring PostgreSQL Data on CVMs

Last updated: 2023-09-10 09:04:52

In the event of data loss or damage, users can utilize the Clone Instance feature to perform point-in-time recovery. The recoverable time is determined by the log retention period.

Downloading Backup in the Console for Restoration

1. Install PostgreSQL Database

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.

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

1. Log in to the Linux-based CVM instance. For more information, see Quickly Configuring Linux CVM.
2. To install the PostgreSQL database, this guide uses the yum repository installation method. You can find the required version at this address.

   > **Note**
   > If you need to restore version 11.8 or 12.4, it is recommended to install the same version of the database. Please change the version number in the installation package, such as modifying `postgresql10-server` to `postgresql11-server` or `postgresql12-server`.

   Run the following command to install PostgreSQL 10:

   ```
   yum install https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg-redh
   yum install postgresql10-server postgresql10-contrib postgresql10 postgresql10.x86_64
   ```

   > **Note**
   > 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-
   > 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 using the postgres user

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

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

In this example, 'recovery' is used as the directory name, but users can modify the recovery directory as needed. In subsequent examples, different version directory names will not be distinguished. Please refer to the actual directory names, such as `/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 Full Backup File

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

> **Note**
> - When downloading a backup using a VPC network address, the TencentDB instance and the CVM instance must be in the same VPC. The backup should be downloaded to the `/var/lib/pgsql/10/recovery` directory.
> - When downloading a backup using a public network address, you need to upload the backup file to the `/var/lib/pgsql/10/recovery` directory on the cloud server. Please refer to How to Copy Local Files to Cloud Server.

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 Excess 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 adding a pound sign (#) at the beginning of the line. If there are multiple instances of the option, comment them all out.

```
shared_preload_libraries
```

```
local_preload_libraries
pg_stat_statements.max
pg_stat_statements.track
archive_mode
archive_command
synchronous_commit
synchronous_standby_names
```

> **Note**
> If the recovery version is PostgreSQL 12.4, you also need to comment out the line `include = 'standby.conf'` .

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_director
```

3. At the end of the `postgresql.conf` file, append the configuration to indicate that the strong synchronization mode will no longer be used.

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

7. Change folder permissions using the 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 database content will be the same as when the full backup was initiated. Place the xlog files in the `/var/lib/pgsql/10/recovery/pg_wal` folder. If the downloaded backup does not

include the `pg_wal` directory, rename the `pg_xlog` directory to `pg_wal`, and PostgreSQL will automatically replay the xlog logs.

For example, if a full backup was taken at 12:00, and all xlogs from 12:00 to 13:00 are placed in the `pg_wal` folder based on that full backup, the database can be restored to the data content at 13:00.

> **Note**
> For PostgreSQL version 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. Extract logs 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 the database using the 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:

```
The command format is: pg_dump -h <Access IP> -U <Access User> -f <Full Path of Backup F
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
```

> **Note**
> As there are extensions 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`.

Deleting Backup

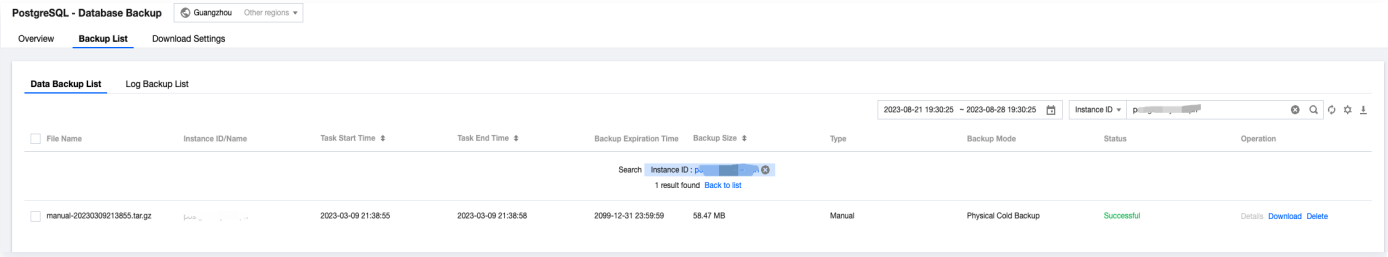
Last updated: 2023-09-10 09:05:01

You can delete database backups to conserve backup storage expenses.

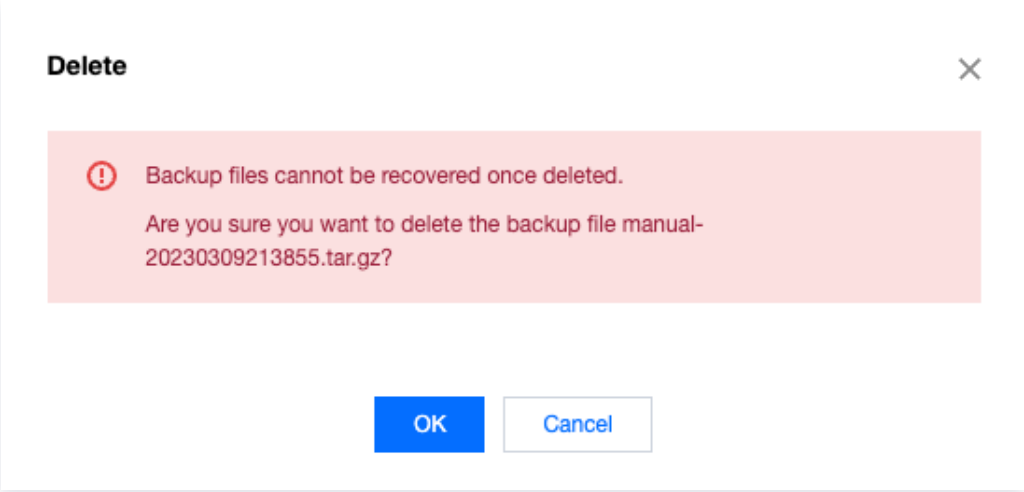
> **Note:**
> Manual backups, automatic backups, and log backups can all be deleted. Once a backup is removed, the data is irretrievable. Specifically, when automatic backups and log backups are deleted, the discontinuity in backup data renders PITR recovery unattainable.

Instructions

1. Log in to the TencentDB for PostgreSQL console, select the region, click the instance ID, and enter the instance management page.
2. On the **Backup and Restoration** tab, locate the backup you want to delete in the backup list, and click **Delete** in the **Operation** column.



3. In the pop-up dialog box, confirm the backup file to be deleted and click **OK**.



Viewing Backup Space

Last updated: 2023-09-10 09:27:35

Scenario

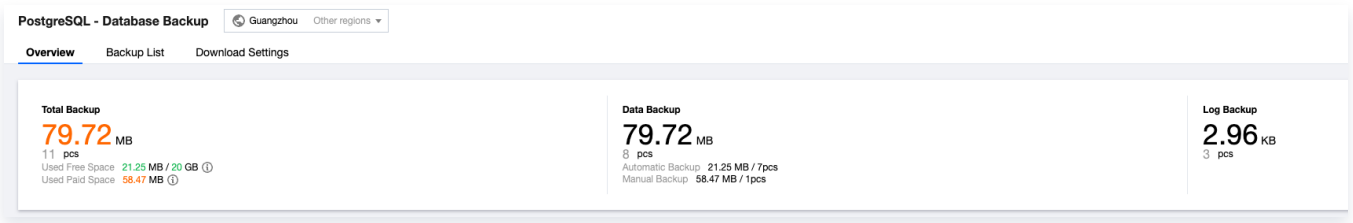
For instances of dual-server high-availability edition, the backup space occupied by TencentDB for PostgreSQL instance backup files is allocated by region. It is equivalent to the total storage capacity used by all database backups in a region, including automatic data backups, manual data backups, and log backups. Increasing manual backup frequency will use more database backup space.

Viewing the backup space

1. Log in to the TencentDB for PostgreSQL console and select **Database Backup** on the left sidebar.
2. Select a region at the top of the page to view its backup information on the **Overview** tab, including total backup and backup statistics.

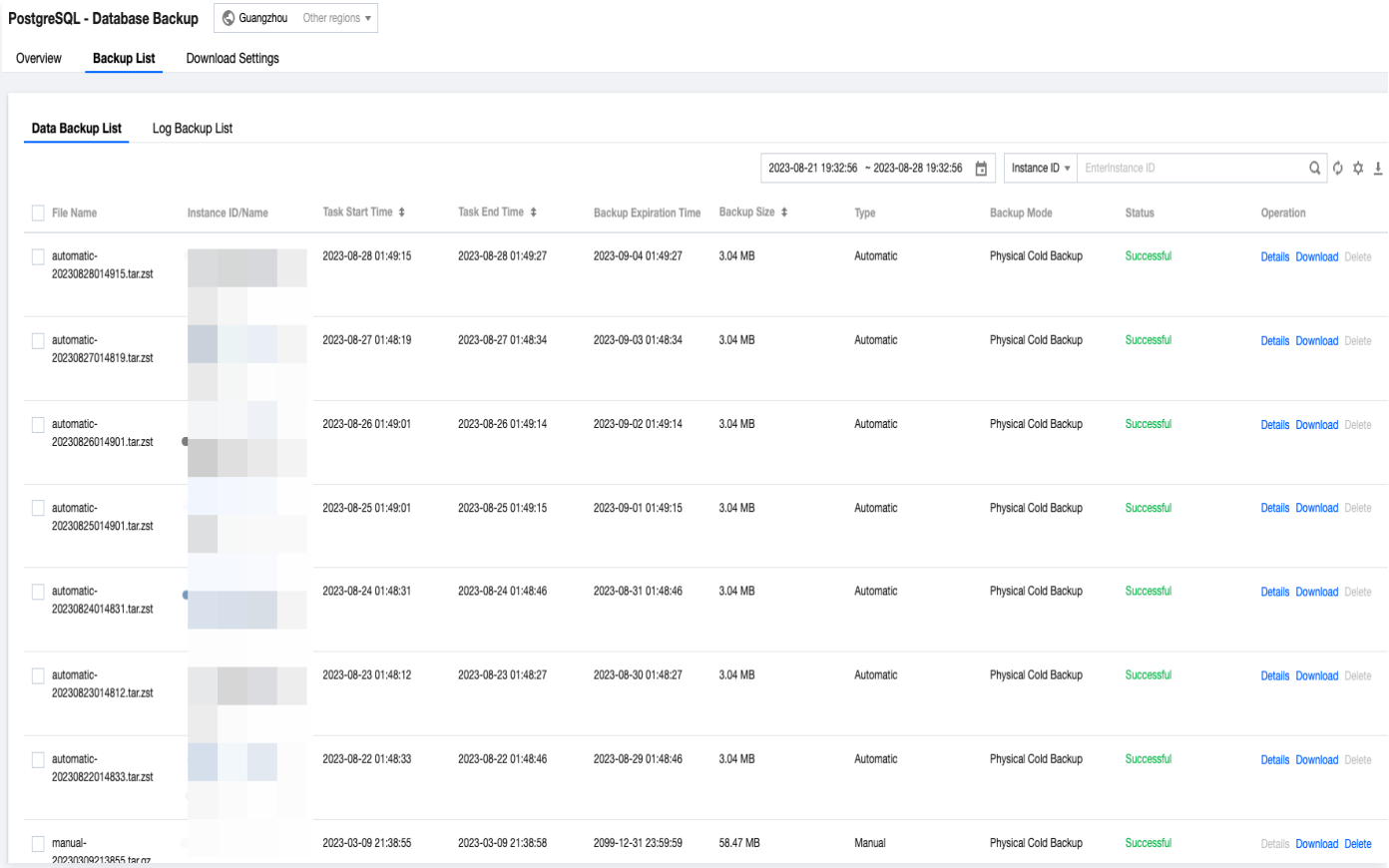
- Total Backup: This section displays the size and quantity of all backups, data backups, and log backups as well as used free/paid space.

> **Note:**
> - Green: The total backup capacity used does not exceed the free tier.
> - Orange: The total backup space used has exceeded the free tier and incurred fees. For more information, see Backup Space Billing.



- Backup Statistics: This section displays the names/IDs/status of instances in the current region, backup space, as well as the size and quantity of data, log, automatic, and manual backups.

3. Select **Backup List** at the top of the page. The backup list is divided into data backups and log backups. Clicking on the instance name in the list will redirect you to the instance details page. The backup list supports filtering by time period, fuzzy search by instance name and ID, and precise search by instance IP.



Data Backup List

- Backups can be sorted by backup time, task start time, task end time, and backup size.
- Click **Details** in the operation column to navigate to the instance backup recovery page. Click **Download** to download the backup.

> **Note:**
> When a monthly subscribed instance or a pay-as-you-go instance is terminated, the system will **provide** you with an additional "final" full physical backup to avoid nonrecovery events due to misoperation. The "final" backup capacity is not included in the backup space statistics, hence no fees are charged. You can download it in the data backup list. The "final" physical backups are automatically deleted seven days after the instance is terminated.

Log Backup List

- Backups can be sorted by log data start time or end time and backup size.
- Click on the details in the operation column to navigate to the instance backup recovery page. Click on **Download** to download the logs, and **Delete** to remove the logs.

> **Note:**
> A log cannot be recovered once deleted. Proceed with caution.

Free Tier

TencentDB for PostgreSQL will start billing for backup space soon. During the beta test of backup billing, the free backup space is the sum of the storage space of all the primary instances in the corresponding region multiplied by 700%. After the backup billing officially starts, the free backup space will be the sum of the storage space of all the primary instances in the corresponding region multiplied by 100%. For more information, see Backup Space Billing.

FAQs

How will I be charged for backup space beyond the free tier? How can I reduce the costs of backup space?

For more information, see Backup Space Billing.

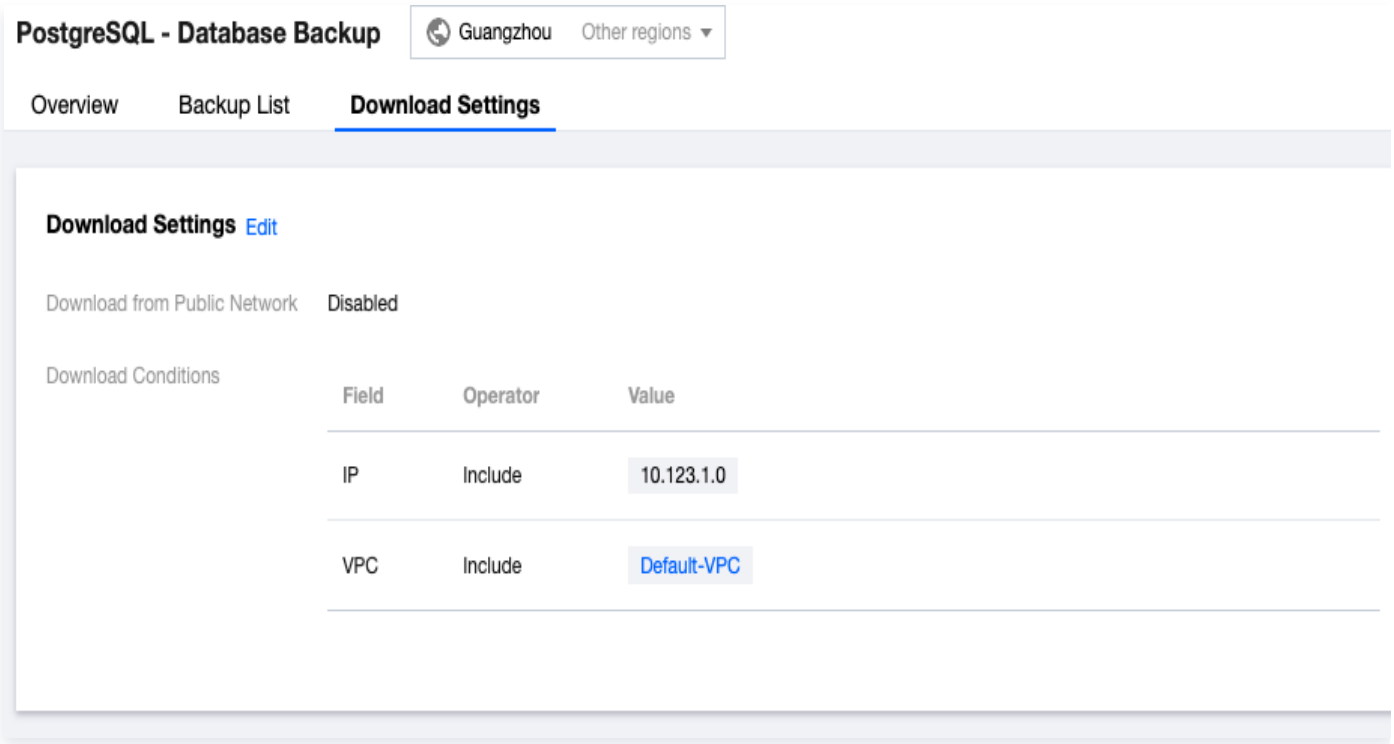
Setting Backup Download Rules

Last updated: 2023-09-10 09:05:17

By default, you can download backup files of TencentDB for PostgreSQL instances over public or private network. To limit the download, you can adjust backup download settings. This document describes how to do so in the console.

1. Log in to the TencentDB for PostgreSQL console and select **Database Backup** on the left sidebar. Then, choose the desired region at the top.
2. On the **Download Settings** tab, view the backup download settings for the selected region and click **Edit** to configure the download restrictions.

> **Note:**
> Download over public network is enabled by default and when it is enabled, download over private network is also allowed.

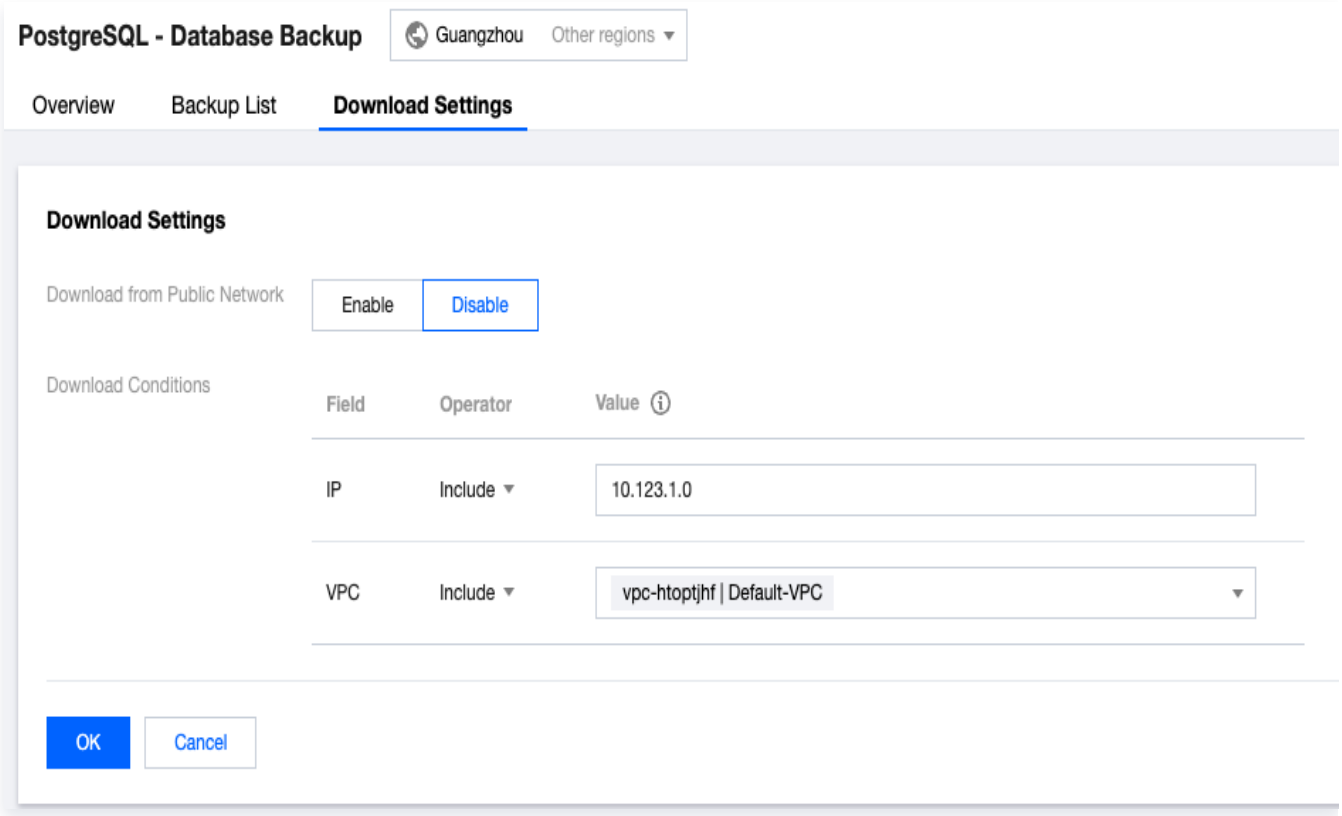


3. In the pop-up window, configure the download settings and click **Confirm**.

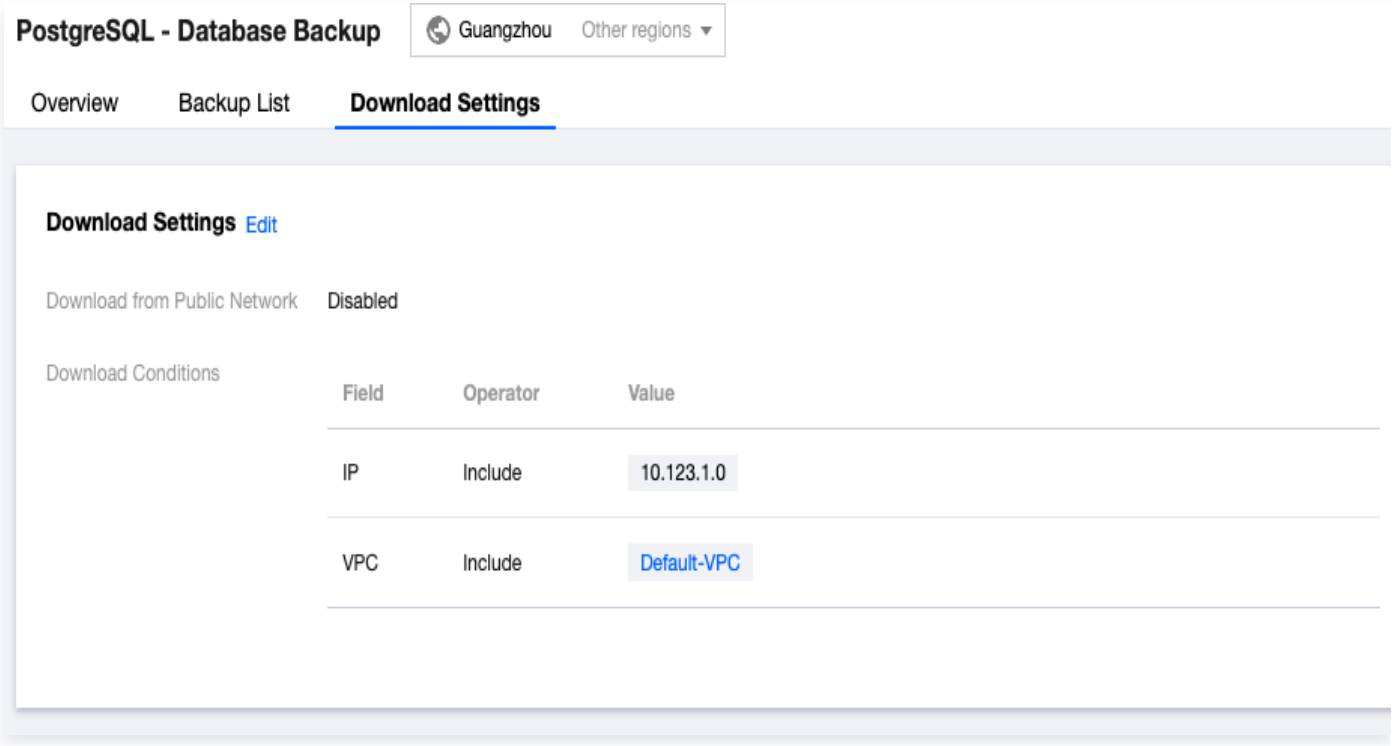
- Download over public network:
  - Enabled: You cannot set any download rule.
  - Disabled: You can set the download rules by allowing or blocking specific IPs and VPCs.
- Set the download rules:
  - If you don't specify any value, the condition won't take effect. If no IP and VPC requirements

are set, there will be no limit on download over private network.

- You should separate the values of an IP condition with commas. You can enter IPs or IP ranges as the values of an IP condition.



4. After completing the configuration, return to the **Download Settings** tab to view the applied restrictions.



Extension Management
Extension Overview

Last updated: 2023-09-10 09:07:37

This document describes the extensions supported by TencentDB for PostgreSQL.

Plugin Overview

TencentDB for PostgreSQL supports multiple open-source and proprietary extensions. They help you perform instance OPS easier and improve the query and write performance as well as various capabilities such as token query, data retrieval, and incremental data migration.

Utilizing Plugins

TencentDB for PostgreSQL supports most commonly used plugins, which can be used directly. However, some plugins require specific versions or special permissions to be enabled. Please submit a ticket and provide the instance ID and plugin name to contact our support staff.

Creating a Plugin

When creating an extension, the `pg_tencentdb_superuser` is temporarily escalated to superuser and passes all permission checks.
TencentDB for PostgreSQL extensions are managed at the database level. You can create different extensions for different databases, but databases cannot use extensions in other databases.
To create an extension, access the database with the client tool and run the following statements:

```
CREATE EXTENSION [ IF NOT EXISTS ] extension_name
[ WITH ]
[ SCHEMA schema_name ]
[ VERSION version ]
[ FROM old_version ]
```

Viewing Created Extension

If you have installed extensions, you can run the following command to view the list of extensions installed in the current database:

- You can run the `\dx` command if you use the psql client.

```
\dx
                                List of installed extensions
  Name   | Version |   Schema   |                     Description
---------+---------+------------+------------------------------------------------------------
 amcheck | 1.2     | public     | functions for verifying relation integrity
 bloom   | 1.0     | public     | bloom access method - signature file based index
```

```
 hstore        | 1.6     | public     | data type for storing sets of (key, value) pairs
 hstore_plperl | 1.0     | public     | transform between hstore and plperl
 jsonb_plperl  | 1.0     | public     | transform between jsonb and plperl
 plperl        | 1.0     | pg_catalog | PL/Perl procedural language
 plpgsql       | 1.0     | pg_catalog | PL/pgSQL procedural language
 postgis       | 3.0.2   | public     | PostGIS geometry, geography, and raster spatial types and fu
(8 rows)
```

- To view the installed plugins using SQL, run the following statement: `select * from pg_available_extensions where installed_version is not null;` .

```
     name      | default_version | installed_version |                        comment
---------------+-----------------+-------------------+-----------------------------------------------------------
 plperl        | 1.0             | 1.0               | PL/Perl procedural language
 amcheck       | 1.2             | 1.2               | functions for verifying relation integrity
 hstore_plperl | 1.0             | 1.0               | transform between hstore and plperl
 plpgsql       | 1.0             | 1.0               | PL/pgSQL procedural language
 jsonb_plperl  | 1.0             | 1.0               | transform between jsonb and plperl
 hstore        | 1.6             | 1.6               | data type for storing sets of (key, value) pairs
 bloom         | 1.0             | 1.0               | bloom access method - signature file based index
 postgis       | 3.0.2           | 3.0.2             | PostGIS geometry, geography, and raster spatial typ
(8 rows)
```

List of Supported Extensions

TencentDB for PostgreSQL supports a variety of powerful and high-performance plugins. For a detailed list of supported plugins for each database version, please refer to Supported Plugins List.

pgAgent Extension

Last updated: 2023-09-10 09:09:02

This document explains how to implement automatic task execution for TencentDB for PostgreSQL using the pgAgent feature. It is recommended to use the pg_cron extension for scheduling tasks.

Feature Overview

If your business needs to perform specified actions in the database at scheduled times, such as clearing redundant data, updating materialized views, performing VACUUM FULL, and executing DML, this can be implemented in PostgreSQL through:

- The crontab feature of Linux
- The pgAgent feature of pgAdmin

pgAgent is a plugin within the pgAdmin tool, introduced in pgAdmin III v1.4. It primarily serves as a job scheduling agent for PostgreSQL, capable of running multi-step batch shell and SQL tasks on complex schedules.
Please note that pgAgent requires support from some database tables and other objects, so you must first install the pgAgent plugin.

Procedure

Configuring pgAgent

1. Log in to your TencentDB for PostgreSQL instance, and after logging in, create your business database.
2. Run the following statement in the database where you need to enable the pgAgent feature and the `postgres` database:

> **Note**
> You must also create pgAgent in the `postgres` database.

```
psql > create extension pgagent;
CREATE EXTENSION
```

3. After configuring the feedback, you need to enable the task scheduler using the pgAgent tool. Log in to CVM (it is recommended that the CVM and PostgreSQL are in the same VPC). Choose the pgAgent version according to the actual database version. In this example, we use version 11.8 and install pgagent_11 from the download link.
4. After pgAgent is installed, run the following statement to start the job scheduler:

> **Note**
> - Use the command based on the actually installed version of pgAgent. For example, if

v10 is installed, the command should be `pgagent_10`.

- Note that `dbname` must be `postgres` rather than the name of the database that needs to run the scheduler; otherwise, the job configuration items will not be displayed on the pgAdmin page.
- If you encounter the error "ERROR: Unsupported schema version" during connection execution, please submit a ticket for assistance.

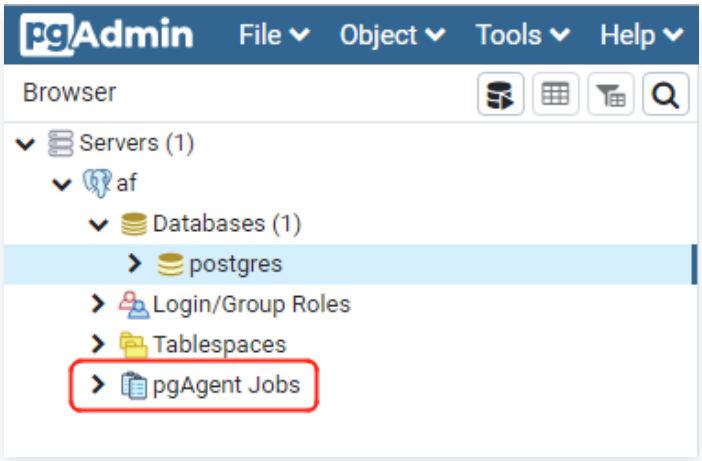
```
pgagent_11 hostaddr=IP dbname=postgres user=username port=port password=password
```

5. After successful execution, there is no echo, but you can use the following command to check whether the process is started successfully:

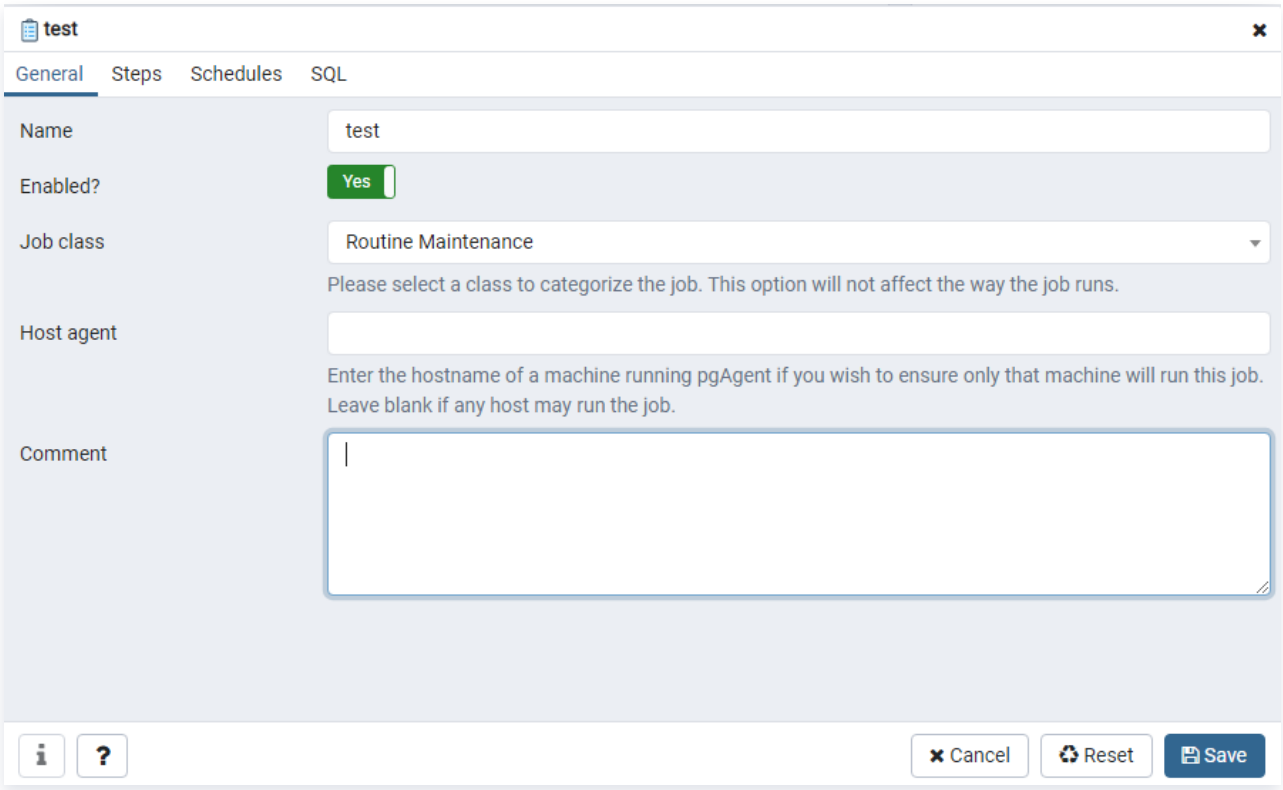
```
Run this statement, and if there is a pgagent process, it has been started successfully.
# ps -ef |grep pgagent
root     158553     1  0 Oct30 ?       00:00:15 pgagent_11 hostaddr=IP dbname=postgres us
```

Configuring pgAgent Jobs through pgAdmin

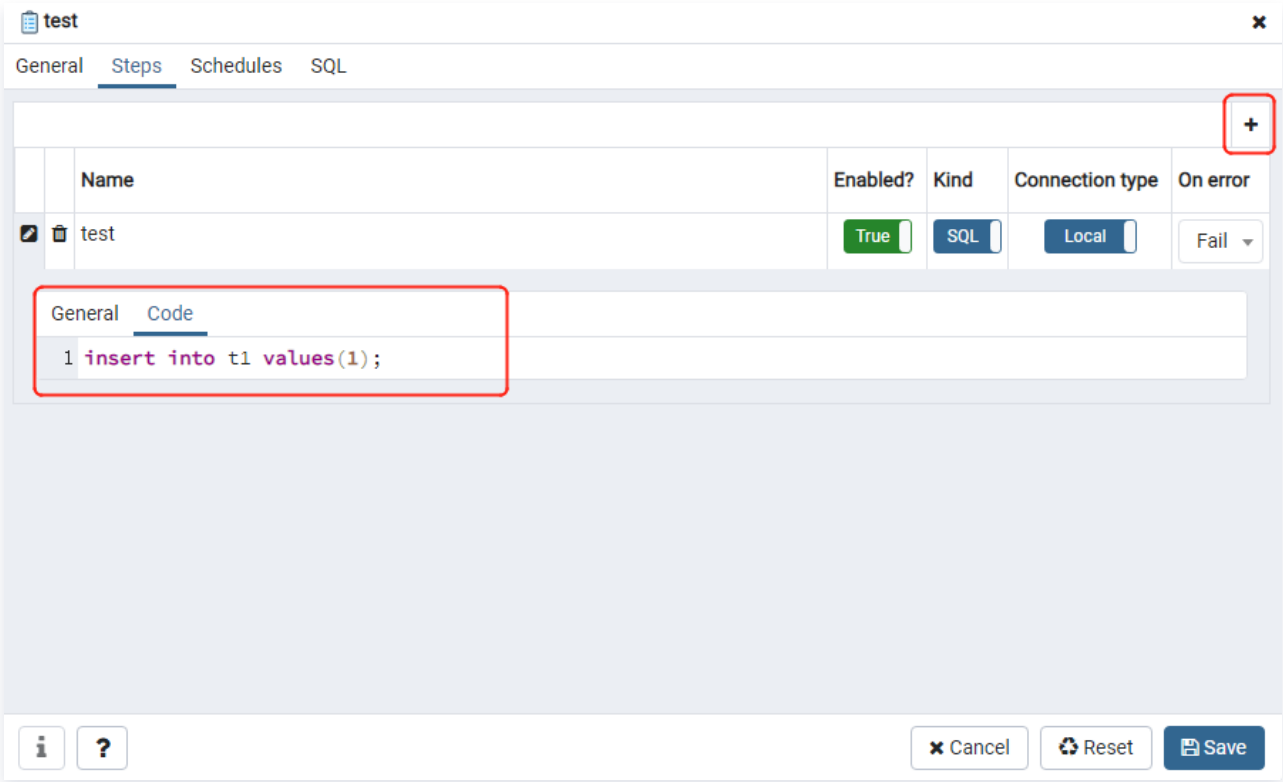
1. Log in to the TencentDB for PostgreSQL console. In the instance list, click an instance ID to enter the instance details page and enable the public network address.
2. Open pgAdmin 4 and access your TencentDB for PostgreSQL instance at the public network access address. At this time, you can see pgAgent Jobs on the page.



3. In the pgAdmin interface, right-click and select **pgAgent Jobs** > **Create** > **Create Jobs** to create a scheduled task.
4. On the **General** page, configure the basic job information.

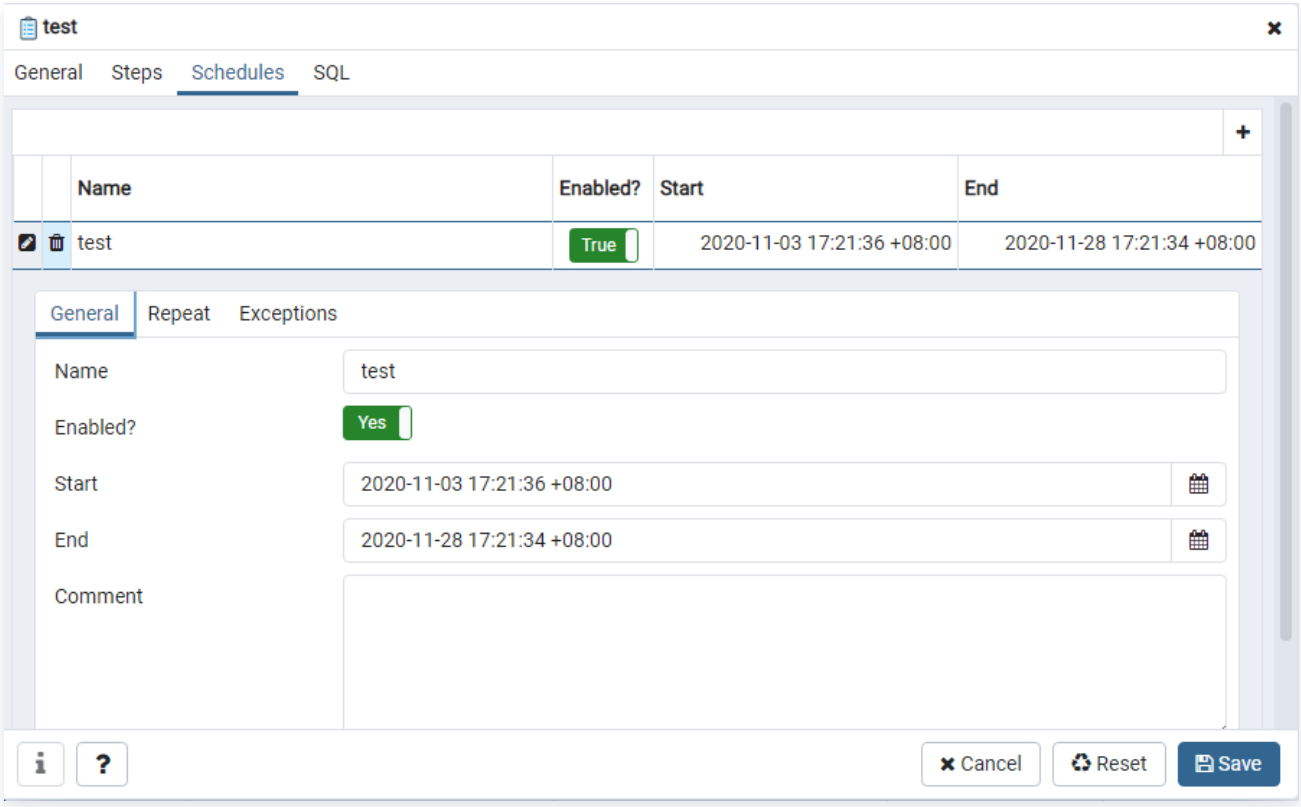


5. Enter the **Steps** tab and configure the job that needs to be executed at the scheduled time. To do so, click + in the top-right corner to add a step, name it, and then configure the SQL statement to be executed on the **Code** tab.

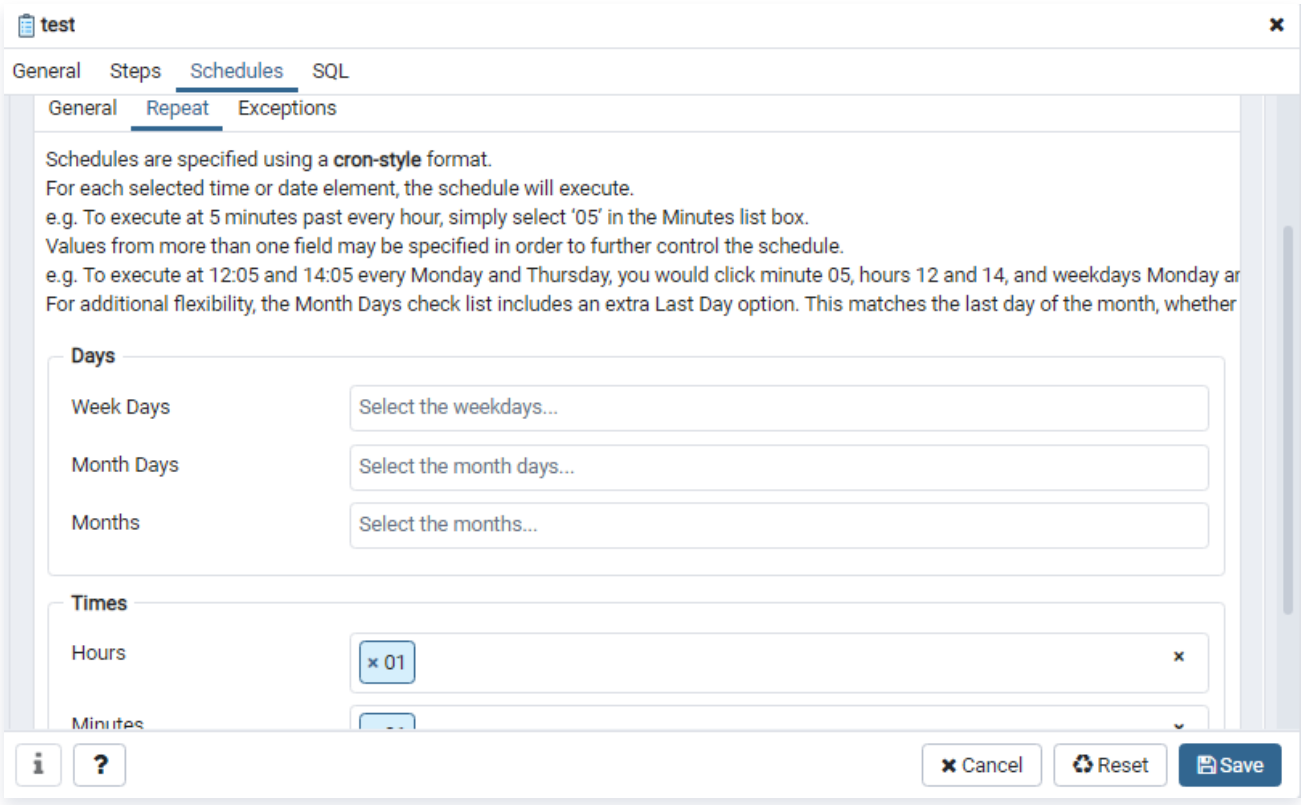


6. Enter the **Schedules** page and configure the scheduling information for job execution:

   6.1 On the **General** tab below, configure the effective time of the job.



6.2 On the **Repeat** tab below, configure a crontab-style schedule.



6.3 After configuring the execution time, you can also configure the time when the job should not be executed on the **Exceptions** tab.

6.4 Click **Save** to complete the process, and the task will be executed automatically according to the configured settings.

postgres_fdw Extension for Cross-database Access

Last updated: 2023-09-10 09:09:11

TencentDB for PostgreSQL provides extensions for accessing external data sources, including other databases in the same instance and other instances. The cross-database access extensions include homogeneous extensions (dblink and postgresql_fdw) and heterogeneous extensions (mysql_fdw and cos_fdw). You can enable cross-database access by following steps below:

1. Install the extensions by running `CREATE EXTENSION`.
2. Create a foreign server object and create link maps for each remote database that needs to be connected.
3. Use the corresponding command to access external tables to get the data.

Cross-database access extensions can directly access across instances or within the same instance for cross-database access. TencentDB for PostgreSQL has optimized permission control when creating foreign server objects, managing them based on the target instance's environment. Additional auxiliary parameters have been added on top of the open-source version to verify user identity and adjust network policies. Please refer to the following section on Plugin Auxiliary Parameters for more details.

> **Note:**
> Please note that the dblink extension is currently only supported by TencentDB for PostgreSQL kernel with major version 10.

Auxiliary Parameters

- **host**
  This parameter is required for cross-instance access. IP address of the target instance
- **port**
  This parameter is required for cross-instance access. Port of the target instance.
- **instanceid**
  Unique instance ID
  - This is used when accessing across instances in TencentDB for PostgreSQL and is mandatory when accessing across instances. The format is similar to postgres-xxxxxx, pgro-xxxxxx, and can be viewed in the Console.
  - If the target instance is a CVM instance, it is the ID of the CVM instance in the format of `ins-xxxxx`.
- **dbname**
  Name of the database in the remote PostgreSQL service to be accessed. For cross-database

access in the same instance, you only need to configure this parameter and can leave other parameters empty.

- **access_type**
  This parameter is optional. Target instance types:
  - 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.
  - The target instance is in a CVM instance.
  - The target instance is a self-built instance with public IP in Tencent Cloud.
  - The target instance is a Tencent Cloud VPN-based instance.
  - The target instance is a self-built VPN-based instance.
  - The target instance is a Direct Connect-based instance.
- **uin**
  This parameter is optional. ID of the account to which the instance belongs, which is used to verify user permissions and can be viewed in Account Info.
- **own_uin**
  This parameter is optional. ID of the root account to which the instance belongs, which is also needed for verifying user permissions.
- **vpcid**
  This parameter is optional. VPC ID. It is required if the target instance is in a CVM instance in a VPC. It can be viewed in the VPC console.
- **subnetid**
  This parameter is optional. Subnet ID for the VPC. It is required if the target instance is in a CVM instance in a VPC. It can be viewed in the VPC console under subnets.
- **dcgid**
  This parameter is optional. Direct Connect connection ID. It is required if the target instance is connected to the network over Direct Connect.
- **vpngwid**
  This parameter is optional. VPN gateway ID. It is required if the target instance is connected to the network over VPN.
- **region**
  This parameter is optional. It indicates the region where the target instance resides; for example, "ap-guangzhou" represents the Guangzhou region. It is required for cross-region access.

postgres_fdw Demo

The `postgres_fdw` extension can be used to access data from other databases in the same instance or other instances.

Step 1. Prepare

1. Create test data in the instance.

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

> **Note**
> If an error occurs during creation, submit a ticket for assistance.

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
```

Step 2. Create the postgres_fdw extension

> **Note**
> If you encounter an error stating that the plugin does not exist or you have insufficient permissions during creation, please submit a ticket for assistance.

```
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)
```

Step 3. Create a server

> **Note**

> Cross-instance access is supported only for kernel v10.17_r1.2, v11.12_r1.2, v12.7_r1.2, v13.3_r1.2, v14.2_r1.0, and later.

- Cross-instance access

```
Access the data of the target instance's testdb2 from the current instance's testdb1
testdb1=>create server srv_test1 foreign data wrapper postgres_fdw options (host 'xxx.xxx.x
CREATE SERVER
```

- For cross-database access in the same instance, you only need to enter the `dbname` parameter.

```
Access the data of testdb2 from testdb1 in the current instance
create server srv_test1 foreign data wrapper postgres_fdw options (dbname 'testdb2');
```

- The target instance is in a CVM instance in the classic network.

```
testdb1=>create server srv_test foreign data wrapper postgres_fdw options (host 'xxx.xxx.x
CREATE SERVER
```

- The target instance is in a CVM instance in a VPC.

```
testdb1=>create server srv_test1 foreign data wrapper postgres_fdw options (host 'xxx.xxx
CREATE SERVER
```

- The target instance is a public network-based self-built instance in Tencent Cloud.

```
testdb1=>create server srv_test1 foreign data wrapper postgres_fdw options (host 'xxx.xxx
CREATE SERVER
```

- The target instance is a Tencent Cloud VPN-based instance.

```
testdb1=>create server srv_test1 foreign data wrapper postgres_fdw options (host 'xxx.xxx
```

- The target instance is a self-built VPN-based instance.

```
testdb1=>create server srv_test1 foreign data wrapper postgres_fdw options (host 'xxx.xxx
```

- The target instance is a Direct Connect-based instance.

```
testdb1=>create server srv_test1 foreign data wrapper postgres_fdw options (host 'xxx.xxx
CREATE SERVER
```

Step 4. Create a user mapping

> **Note**
> You can skip this step for cross-database access in the same instance.

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

Step 5. Create a foreign table

```
testdb1=> create foreign table foreign_table1(id integer) server srv_test1 options(table_name 't
CREATE FOREIGN TABLE
```

Step 6. Access data from foreign table

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

Reference

- Official Introduction to postgres_fdw
- Create a server on v9.3
- Create a server on v9.5
- Create a server on v10
- Create a server on v11
- Create a server on v12
- Create a server on v13
- Create a server on v14

Sample for Using `dblink`

Step 1. Create the `dblink` extention

```
postgres=> create extension dblink;
postgres=> \dx
                                 List of installed extensions
      Name       | Version |   Schema   |                     Description
-----------------+---------+------------+---------------------------------------------------------------
 dblink          | 1.2     | public     | connect to other PostgreSQL databases from within a databas
```

```
 pg_stat_log        | 1.0     | public     | track runtime execution statistics of all SQL statements exe
 pg_stat_statements | 1.6     | public     | track execution statistics of all SQL statements executed
 plpgsql            | 1.0     | pg_catalog | PL/pgSQL procedural language
(4 rows)
```

Step 1. Create the `dblink` link

```
select dblink_connect('yunpg1','host=10.10.10.11 port=5432 instanceid=postgres-2123455r db
 dblink_connect
----------------
 OK
(1 row)
```

Step 3 Access external data

```
postgres=> select * from dblink('yunpg1','select catalog_name,schema_name,schema_owner fr
   a    |        b         |    c
----------+------------------+---------
 postgres | pg_toast         | user_00
 postgres | pg_temp_1        | user_00
 postgres | pg_toast_temp_1  | user_00
 postgres | pg_catalog       | user_00
 postgres | public           | user_00
 postgres | information_schema | user_00
(6 rows)
```

Reference

dblink Overview

Sample for Using `mysql_fdw`

Step 1. Create the `mysql_fdw` extension

```
postgres=> create extension mysql_fdw;
CREATE EXTENSION
postgres=> \dx;
                                  List of installed extensions
      Name       | Version |   Schema   |                     Description
-------------------+---------+------------+-----------------------------------------------------------------------
 dblink           | 1.2     | public     | connect to other PostgreSQL databases from within a databas
 mysql_fdw        | 1.1     | public     | Foreign data wrapper for querying a MySQL server
 pg_stat_log      | 1.0     | public     | track runtime execution statistics of all SQL statements exe
 pg_stat_statements | 1.9     | public     | track planning and execution statistics of all SQL statem
 plpgsql          | 1.0     | pg_catalog | PL/pgSQL procedural language
(5 rows)
```

Step 2. Create a server

```
postgres=> CREATE SERVER mysql_svr  FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host '17:
CREATE SERVER
```

Step 3. Create a user mapping

```
postgres=> CREATE USER MAPPING FOR PUBLIC SERVER mysql_svr OPTIONS (username 'fdw_us
CREATE USER MAPPING
```

Step 4. Access external data

```
postgres=> IMPORT FOREIGN SCHEMA hrdb FROM SERVER mysql_svr INTO public;
```

Reference

mysql_fdw Official Overview

Sample for Using `cos_fdw`

For examples of using cos_fdw, please refer to the document Supporting Hierarchical Storage Capability through cos_fdw Plugin.

Note

Pay attention to the following for the target instance:

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, see PostgreSQL's official documentation.
2. If the target instance is not a TencentDB instance and has a hot backup mode configured, after a primary-standby switch, you need to update the server connection address or create a server again.

pg_roaringbitmap Extension for Bitwise Operation

Last updated: 2023-09-10 09:31:03

TencentDB for PostgreSQL provides the `pg_roaringbitmap` extension to use the bitwise operation feature to improve the query performance.

Preparations

Your TencentDB for PostgreSQL instance is on v10, 11, 12, or 13.

Background Information

The Roaring Bitmap algorithm divides 32-bit INT data into 216 chunks, with each chunk corresponding to the high 16 bits of an integer and using a container to store the low 16 bits of a value. Roaring Bitmap stores these containers in a dynamic array as a first-level index. The containers use two different structures: Array Container and Bitmap Container. Array Containers store sparse data, while Bitmap Containers store dense data. If the number of integers in a container is less than 4,096, an Array Container is used to store the values. If it is greater than 4,096, a Bitmap Container is used.

By adopting this storage structure, Roaring Bitmap can quickly retrieve a specific value. When performing bitwise operations (AND, OR, XOR), Roaring Bitmap provides corresponding algorithms to efficiently implement operations between the two types of containers. This ensures that Roaring Bitmap performs excellently in both storage and computation.

Instructions

1. Create the plugin. Below is an example:

```
CREATE EXTENSION roaringbitmap;
```

2. Create a table with the RoaringBitmap data type. An example is shown below:

```
CREATE TABLE t1 (id integer, bitmap roaringbitmap);
```

3. Use the `rb_build` function to insert data into Roaring Bitmap. Here's an example:

```
-- Set the bit value of the array to 1.
INSERT INTO t1 SELECT 1,RB_BUILD(ARRAY[1,2,3,4,5,6,7,8,9,200]);
-- Set the bit values of multiple records to 1 and aggregate the bit values into a Roaring bitma
INSERT INTO t1 SELECT 2,RB_BUILD_AGG(e) FROM GENERATE_SERIES(1,100) e;
```

4. Perform Bitmap calculations (OR, AND, XOR, ANDNOT). Below is an example:

```
-- Set the bit value of the array to 1.
SELECT RB_OR(a.bitmap,b.bitmap) FROM (SELECT bitmap FROM t1 WHERE id = 1) AS a,(SELE
```

5. Perform Bitmap aggregation operations (OR, AND, XOR, BUILD) and generate a new `roaringbitmap` type. The following example demonstrates this:

```
SELECT RB_OR_AGG(bitmap) FROM t1;
SELECT RB_AND_AGG(bitmap) FROM t1;
SELECT RB_XOR_AGG(bitmap) FROM t1;
SELECT RB_BUILD_AGG(e) FROM GENERATE_SERIES(1,100) e;
```

6. Cardinality refers to counting the number of BIT positions set to 1 in a roaringbitmap. Here's an example:

```
SELECT RB_CARDINALITY(bitmap) FROM t1;
```

7. Retrieve the BIT index at position 1 from the `roaringbitmap` (i.e., the position value). The example is as follows:

```
SELECT RB_ITERATE(bitmap) FROM t1 WHERE id = 1;
```

Feature Function List

Function Name	Input	Output	Description	Sample	Result
rb_build	integer[]	roaringbitmap	Create roaringbitmap from integer array	`` `rb_build('{1,2,3,4,5}')` ``	{1,2,3,4,5}
rb_index	roaringbitmap,integer	bigint	Return the 0-based index of element in this roaringbitmap, or -1 if do	`` `rb_index('{1,2,3}',3)` ``	2

			not exsits		
rb_cardinality	roaringbitmap	bigint	Return cardinality of the roaringbitmap	`rb_cardinality('{1,2,3,4,5}')`	5
rb_and_cardinality	roaringbitmap,roaringbitmap	bigint	Return cardinality of the AND of two roaringbitmaps	`rb_or_cardinality('{1,2,3}','{3,4,5}')`	1
rb_xor_cardinality	roaringbitmap,roaringbitmap	bigint	Return cardinality of the XOR of two roaringbitmaps	`rb_xor_cardinality('{1,2,3}','{3,4,5}')`	4
rb_andnot_cardinality	roaringbitmap,roaringbitmap	bigint	Return cardinality of the ANDNOT of two roaringbitmaps	`rb_andnot_cardinality('{1,2,3}','{3,4,5}')`	2
rb_is_empty	roaringbitmap	boolean	Check if roaringbitmap is empty.	`rb_is_empty('{1,2,3,4,5}')`	t
rb_fill	roaringbitmap,range_start bigint,range_end bigint	roaringbitmap	Fill the specified range (not include the range_end)	`rb_fill('{1,2,3}',5,7)`	{1,2,3,5,6}
rb_clear	roaringbitmap,range_start	roaringbitmap	Clear the		{1,3}

	bigint,range_end bigint		specified range (not include the range_end)	`rb_clear('{1,2,3}',2,3)`	
rb_flip	roaringbitmap,range_start bigint,range_end bigint	roaringbitmap	Negative the specified range (not include the range_end)	`rb_flip('{1,2,3}',2,10)`	{1,4,5,6,7,8,9}
rb_range	roaringbitmap,range_start bigint,range_end bigint	roaringbitmap	Return new set with specified range (not include the range_end)	`rb_range('{1,2,3}',2,3)`	{2}
rb_range_cardinality	roaringbitmap,range_start bigint,range_end bigint	bigint	Return the cardinality of specified range (not include the range_end)	`rb_range_cardinality('{1,2,3}',2,3)`	1
rb_min	roaringbitmap	integer	Return the smallest offset in roaringbitmap. Return NULL if	`rb_min('{1,2,3}')`	1

			the bitmap is empty		
rb_max	roaringbitmap	integer	Return the greatest offset in roaringbitmap. Return NULL if the bitmap is empty	`rb_max('{1,2,3}')`	3
rb_rank	roaringbitmap,integer	bigint	Return the number of elements that are smaller or equal to the specified offset	`rb_rank('{1,2,3}',3)`	3
rb_jaccard_dist	roaringbitmap,roaringbitmap	double precision	Return the jaccard distance (or the Jaccard similarity coefficient) of two bitmaps	`rb_jaccard_dist('{1,2,3}','{3,4}')`	0.25
rb_select	roaringbitmap,bitset_limit bigint,bitset_offset bigint=0,reverse boolean=false,range_start bigint=0,range_en	roaringbitmap	Return subset [bitset_offset,bitset_offset+bitset_limit) of bitmap	`rb_select('{1,2,3,4,5,6,7,8,9}',5,2)`	{3,4,5,6,7}

	d bigint=4294967296		between range [range_start,range_end)		
rb_to_array	roaringbitmap	integer[]	Convert roaringbitmap to integer array	`rb_to_array(roaringbitmap('{1,2,3}'))`	{1,2,3}
rb_iterate	roaringbitmap	SET of integer	Return set of integer from a roaringbitmap data.	`SELECT rb_iterate(rb_build('{1,2,3}'))`	1 2 3

Aggregate Function List

Aggregate Function Name	Input	Output	Description	Sample	Result
rb_build_agg	integer	roaringbitmap	Build a roaringbitmap from a integer set	select rb_build_agg(id) from (values (1),(2),(3)) t(id)	{1,2,3}
rb_or_agg	roaringbitmap	roaringbitmap	AND Aggregate calculations from a roaringbitmap set	select rb_or_agg(bitmap) from (values (roaringbitmap('{1,2,3}')),(roaringbitmap('{2,3,4}'))) t(bitmap)	{1,2,3,4}
rb_and_agg	roaringbitmap	roaringbitmap	AND Aggregate calculations from a roaringbitmap set	select rb_and_agg(bitmap) from (values (roaringbitmap('{1,2,3}')),(roaringbitmap('{2,3,4}'))) t(bitmap)	{2,3}
rb_xor_agg	roaringbitmap	roaringbitmap	XOR Aggregate calculations	select rb_xor_agg(bitmap) from (values	{1,4}

			from a roaringbitmap set	(roaringbitmap('{1,2,3}')), (roaringbitmap('{2,3,4}'))) t(bitmap)	
rb_or_cardinality_agg	roaringbitmap	bigint	OR Aggregate calculations from a roaringbitmap set, return cardinality.	select rb_or_cardinality_agg(bitmap) from (values (roaringbitmap('{1,2,3}')), (roaringbitmap('{2,3,4}'))) t(bitmap)	4
rb_and_cardinality_agg	roaringbitmap	bigint	AND Aggregate calculations from a roaringbitmap set, return cardinality	select rb_and_cardinality_agg(bitmap) from (values (roaringbitmap('{1,2,3}')), (roaringbitmap('{2,3,4}'))) t(bitmap)	2
rb_xor_cardinality_agg	roaringbitmap	bigint	XOR Aggregate calculations from a roaringbitmap set, return cardinality	select rb_xor_cardinality_agg(bitmap) from (values (roaringbitmap('{1,2,3}')), (roaringbitmap('{2,3,4}'))) t(bitmap)	2

pg_cron Extension for Job Scheduling

Last updated: 2023-09-10 09:31:53

pg_cron is a simple cron-based task scheduler for PostgreSQL (version 10 or higher) that runs as an extension within the database. It utilizes the same syntax as regular cron, enabling you to schedule and execute database commands directly from the database at specified intervals.
This document describes how to use the pg_cron extension of PostgreSQL.

Enabling pg_cron Extension

1. To use pg_cron, first submit a ticket to have pg_cron added to the database's shared_preload_libraries parameter. Modifying this parameter requires restarting the instance, so please choose a period of low business activity to perform this operation.
2. After the parameter is modified, enter the `postgres` database and run the following command with the admin account:

```
CREATE EXTENSION pg_cron;
```

3. Currently, pg_cron is configured to execute scheduled tasks only in the `postgres` database. If you need to run scheduled tasks in other databases, please refer to Setting up Scheduled Tasks for Databases other than Postgres.
4. By default, after pg_cron is created, its configuration data and task execution can only be set by the admin user. If you need other users to set up or execute pg_cron, you must grant them permissions to the cron metadata database. Run the following command:

```
postgres=> GRANT USAGE ON SCHEMA cron TO other-user;
```

This permission grants the `other-user` access to the cron metadata for scheduling and unscheduling cron tasks. However, to successfully run cron tasks, access to the objects within the cron task is also required. If the user lacks the necessary permissions, the task will fail, and an error will be displayed in the `postgresql.log`.
In the following example, the user does not have access to the `pgbench_accounts` table.

```
2020-12-08 16:41:00 UTC::@:[30647]:ERROR: permission denied for table pgbench_accounts
2020-12-08 16:41:00 UTC::@:[30647]:STATEMENT: update pgbench_accounts set abalance =
2020-12-08 16:41:00 UTC::@:[27071]:LOG: background worker "pg_cron" (PID 30647) exited
```

Below are other messages in the `cron.job_run_details` table:

```
postgres=> select jobid, username, status, return_message, start_time from cron.job_run_del
jobid |  username  | status |                return_message                |         start_time
```

```
-------+------------+--------+-----------------------------------------------------+-----------------------------
143 | unprivuser | failed | ERROR: permission denied for table pgbench_accounts | 2020-12-08
143 | unprivuser | failed | ERROR: permission denied for table pgbench_accounts | 2020-12-08
143 | unprivuser | failed | ERROR: permission denied for table pgbench_accounts | 2020-12-08
143 | unprivuser | failed | ERROR: permission denied for table pgbench_accounts | 2020-12-08
143 | unprivuser | failed | ERROR: permission denied for table pgbench_accounts | 2020-12-08
(5 rows)
```

pg_cron Scheduled Job Configuration

pg_cron provides three main operations: adding and deleting jobs and viewing job information.

cron.schedule() function

This function schedules cron tasks. Tasks are initially scheduled in the default `postgres` database. The function returns a bigint value representing the task identifier. To schedule tasks to run in other databases within the PostgreSQL instance, refer to the example in Scheduling Tasks for Databases Other Than postgres.

There are two syntax formats for this function.

Syntax

```
cron.schedule (job_name,
   schedule,
   command
);

cron.schedule (schedule,
   command
);
```

Parameter

Category	Description
job_name	The name of the cron task, which can be left empty or unset.
schedule	Text representing the cron task schedule in standard cron format.
command	Text of the command to be executed.

Example

```
postgres=> SELECT cron.schedule ('test','0 10   *', 'VACUUM pgbench_history');
 schedule
----------
      145
(1 row)
```

```
postgres=> SELECT cron.schedule ('0 15   *', 'VACUUM pgbench_accounts');
 schedule
----------
      146
(1 row)
```

`schedule` uses the standard cron syntax. Here, `*` indicates to run the job at the specified time, and specific numbers indicate to run the job at the time specified by the numbers.

```
# Format: minute  hour  day of month  month  day of week
# week (0 - 6) = sun,mon,tue,wed,thu,fri,sat
# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,...,sat
# |  |  |  |  |
#        *
```

cron.unschedule() function

This function removes a cron task. You can pass either the job_name or job_id. Ensure that you are the owner of the policy corresponding to the current job_id. The function returns a boolean value, indicating success or failure.
The function uses the following syntax format.

Syntax

```
cron.unschedule (job_id);

cron.unschedule (job_name);
```

Parameter

Category	Description
job_id	The task identifier returned from the cron.schedule function when scheduling a cron job.
job_name	Name of the cron task scheduled using the cron.schedule function.

Example

```
postgres=> select cron.unschedule(108);
 unschedule
```

```
------------
 t
(1 row)

postgres=> select cron.unschedule('test');
 unschedule
------------
 t
(1 row)
```

pg_cron tables

The following tables are used to schedule jobs and record job execution methods.

Table	Description
cron.job	This table contains metadata about each scheduled task. Most interactions with this table should be done using the cron.schedule and cron.unschedule functions. Note: It is not recommended to grant direct update or insert permissions to this table.
cron.job_run_details	This includes historical information of past scheduled tasks. It is useful for investigating the status, return messages, and start and end times of executed tasks. To prevent this table from growing indefinitely, please perform regular maintenance to clear it.

Setting pg_cron Scheduled Job

1. If you wish to schedule a vacuum operation on a specific table at a chosen time, you can use the cron.schedule function to set up a timed task, such as performing VACUUM FREEZE on a specific table every day at 22:00 (GMT). The number returned by the setup statement represents the task ID for the current task.

```
SELECT cron.schedule('manual vacuum', '0 22   *', 'VACUUM FREEZE pgbench_accounts');
 schedule
----------
1
(1 row)
```

2. This function has three input parameters: the job name (string), the cron scheduling syntax, and the specific SQL statement to be executed.

Viewing pg_cron Scheduled Job

After scheduling a job, you can view it in the `cron.job` table by running the following statement:

```
SELECT * FROM cron.job;

jobid | schedule   |  command  | nodename  | nodeport | database | username | active
-------+------------+-----------+-----------+----------+----------+----------+--------
    1 | 0 22   * |   VACUUM ... | localhost |     5432 | postgres | test     | t
```

Deleting pg_cron Scheduled Job

If a scheduled task is no longer needed, it should be deleted. Execute the following statement to remove the scheduled task:

```
SELECT cron.unschedule(1);

 unschedule
------------
          t
```

Viewing the Execution History of Scheduled Job

After running the above sample code, you can check the job status and execution result in the `cron.job_run_details` table as follows:

```
postgres=> select * from cron.job_run_details;
 jobid | runid | job_pid | database | username | command | status | return_message | start_time |
-------+-------+---------+----------+----------+-----------------------------------------+-----------+----------------+------
 1 | 1 | 3395 | postgres | adminuser| vacuum freeze pgbench_accounts | succeeded | VACUUM | 2
(1 row)
```

Clearing pg_cron Record Table

1. The `cron.job_run_details` table contains the history of cron jobs, and over time, these records may become quite large. We recommend that you plan to clean up the records in this table. For example, retaining a week's worth of entries may be sufficient for troubleshooting purposes.
2. The following example uses the cron.schedule function to schedule a daily midnight run to clear records from the cron.job_run_details table, retaining only the past seven days of historical records, as shown below.

```
SELECT cron.schedule('0 0   *', $$DELETE
   FROM cron.job_run_details
   WHERE end_time < now() - interval '7 days'$$);
```

Disabling pg_cron History

– To completely disable writing any content into the `cron.job_run_details` table, set the `cron.log_run` parameter to `off` in the console.
Upon executing this operation, the pg_cron extension will no longer write to tables and will only generate errors in the postgresql.log file. All error information can be viewed in the error logs within the console.
– Run the following command to check the value of the `cron.log_run` parameter.

```
postgres=> SHOW cron.log_run;
```

Scheduling Tasks for Databases Other Than postgres

By default, pg_cron's metadata is stored entirely in the `postgres` database. If you need to execute scheduled tasks on objects in other databases, perform the following steps:

1. To perform the `VACUUM` operation on a table in the `test` database, you first need to use the admin account of pg_cron to run the `cron.schedule` function in the `postgres` database to schedule a job.

```
postgres=> SELECT cron.schedule('test manual vacuum', '29 03   *', 'vacuum freeze test_table')
```

2. As an admin user, execute the database change to set the scheduled task execution `database` to the desired database. Please note that the `jobid` must be the same as the one returned in the first step. Use the following command:

```
postgres=> UPDATE cron.job SET database = 'test' WHERE jobid = 106;
```

3. Query the `cron.job` table to verify the operation result.

```
postgres=> select * from cron.job;
 jobid | schedule | command | nodename | nodeport | database | username | active | jobname
-------+------------+-----------------------------------+-----------+----------+----------+----------+--------+-----
 2 | 29 03   * | vacuum freeze test_table | localhost | 8192 | test | adminuser | t | database1 manu
 1 | 59 23   * | vacuum freeze pgbench_accounts | localhost | 8192 | postgres | adminuser | t | ma
(2 rows)
```

pg_cron Parameters

Parameter used to control the behaviors of the pg_cron extension are as listed below:

Category	Description
cron.database_name	Storing pg_cron Metadata in the Database.
cron.host	The hostname to connect to PostgreSQL. You cannot modify this value.

cron.log_run	Record all running tasks in the `job_run_details` table. The value can be either `on` or `off`.
cron.log_statement	Log all cron statements before executing them. The value can be set to 'on' or 'off'.
cron.max_running_jobs	The maximum number of tasks that can run concurrently. If you need support for more tasks, please submit a ticket for assistance.
cron.use_background_workers	Using a background worker instead of a client session. You cannot modify this value.

You can run the following SQL command to display these parameters and their values:

```
postgres=> SELECT name, setting, short_desc FROM pg_settings WHERE name LIKE 'cron.%' OR
```

Network Management Overview

Last updated: 2023-09-10 09:09:42

TencentDB for PostgreSQL offers network management to ensure the security, efficiency, and flexibility of your instances when providing services both internally and externally. This document introduces the network-related information for PostgreSQL.

Network

There are two types of TencentDB network environments: VPC and classic network.

- VPC: A Virtual Private Cloud is a logically isolated network space that you can customize in Tencent Cloud. By default, different VPCs cannot communicate with each other, even within the same region. Similar to a traditional network in a data center, a VPC in Tencent Cloud hosts your cloud service resources.
- Classic network: This is a public network resource pool for all Tencent Cloud users. Tencent Cloud centrally manages all cloud resources for users, making management simple and efficient.

> **Note**
> Currently, resources cannot be created in the classic network.

Feature differences

SDK	Classic network	VPCs
Network Customization	Unavailable	This feature is supported.
Custom Route Configuration	Unavailable	This feature is supported.
Custom IP	Unavailable	This feature is supported.
Intercommunication rules	Intra-region connectivity	Interconnectivity between subnets within the same VPC and region.
Security control	Security Group	Security Group

Network access

Tencent Cloud services can be accessed over both the public and private networks.

- Public network access: This is a service provided by Tencent Cloud for instances to perform public data transfer. Instances enable public IP addresses to communicate with other computers on the network or to access instances through the public network.
- Private network access: it is used to provide Local Area Network (LAN) service. Tencent Cloud

assigns resources with private IP addresses to allow a free private network communication in the same region or instance access over the private network.

> **Note**
> Security groups that currently support public network access are available only in the Guangzhou, Shanghai, Beijing, and Chengdu regions. Instances in other regions may be attacked if the public network access is enabled. We do not recommend that instances in production environment enable public network access. If you need to enable public network access, security group rules must be configured.

Network Configuration

You can configure one or two networks for each TencentDB for PostgreSQL instance.
In scenarios where the instance supports two networks:

- An instance can be accessed through different VIPs that belong to different VPCs and subnets.
- You can use this feature to change the instance network, for example, from the classic network to VPC or from VPC A to VPC B.
- You can use this feature to implement the multi-plane network feature in scenarios where businesses in two different VPCs need to access the same database instance.

Managing Instance Network

The TencentDB for PostgreSQL console provides management operations for the instance's network, including adding, deleting, and modifying networks. For more information, see Modify Network.

Modifying Network

Last updated: 2023-09-10 09:09:53

You can add, delete, or modify the network configuration of instances based on your business requirements. This document introduces how to configure and manage the network of TencentDB for PostgreSQL instances through the console.

Scenario

Tencent Cloud provides the **classic network** and **Virtual Private Cloud (VPC)** for different scenarios. Based on these, we offer the following flexible services to help you configure and manage your networks.

- **Network Change**
- Switch from classic network to VPC: a single TencentDB source instance can be switched from classic network to VPC.
- Switch from VPC A to VPC B: a single primary or read-only TencentDB instance can be switched from VPC A to VPC B.
- **Customize Access IP Address**
- Custom primary instance IP: you can specify the IP address when adding a network on the instance details page of the primary instance.
- Custom read-only instance IP: you can specify the IP address when adding a network on the instance details page of a read-only instance.

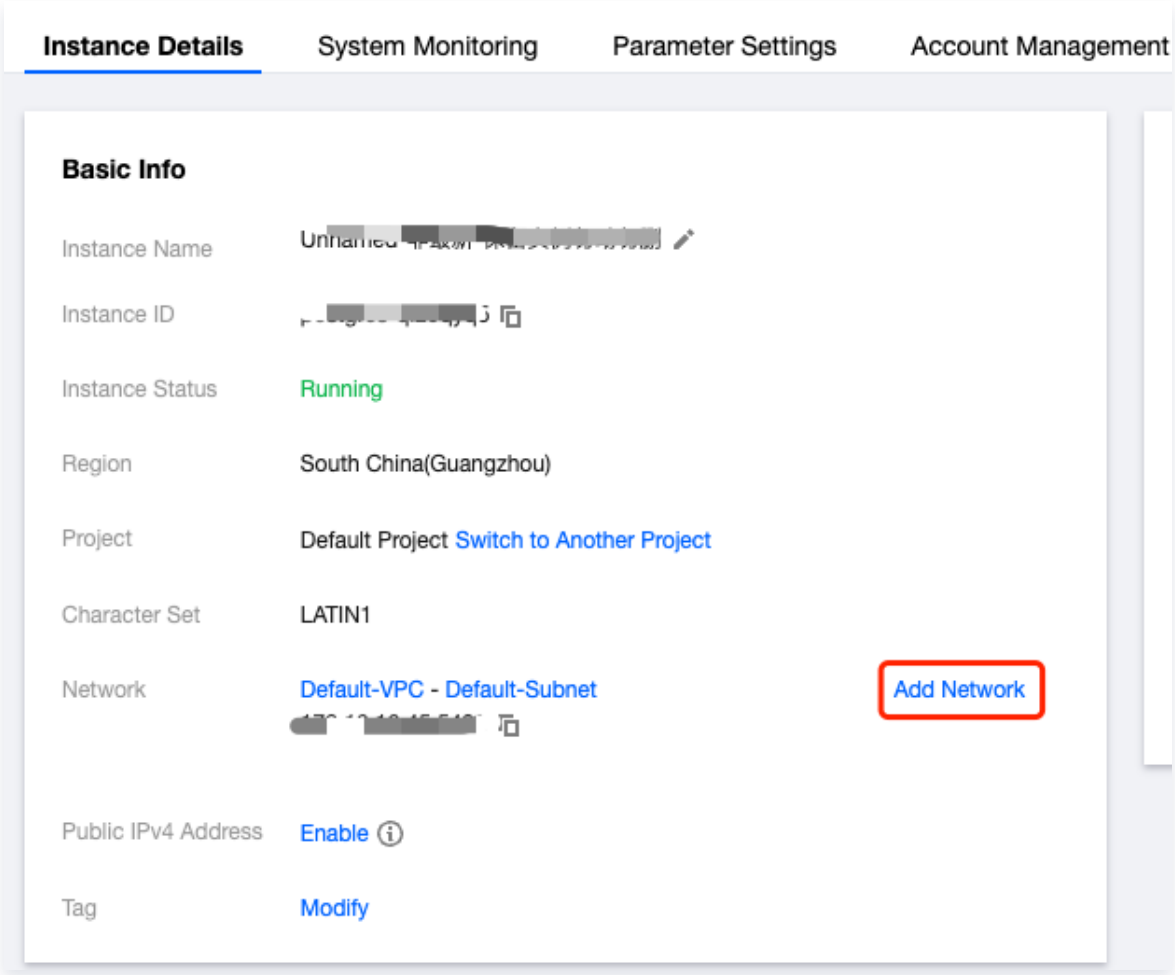
Supports and Limits

- The change from classic network to VPC is irreversible. After the switch to a VPC, the TencentDB instance cannot communicate with Tencent Cloud services in another VPC or classic network.
- After you change a primary instance's network, the networks of read-only instances associated with the primary instance won't be automatically switched; that is, you need to separately switch them.
- A new network added to an instance does not affect the IP address in the original network configuration.

Instructions

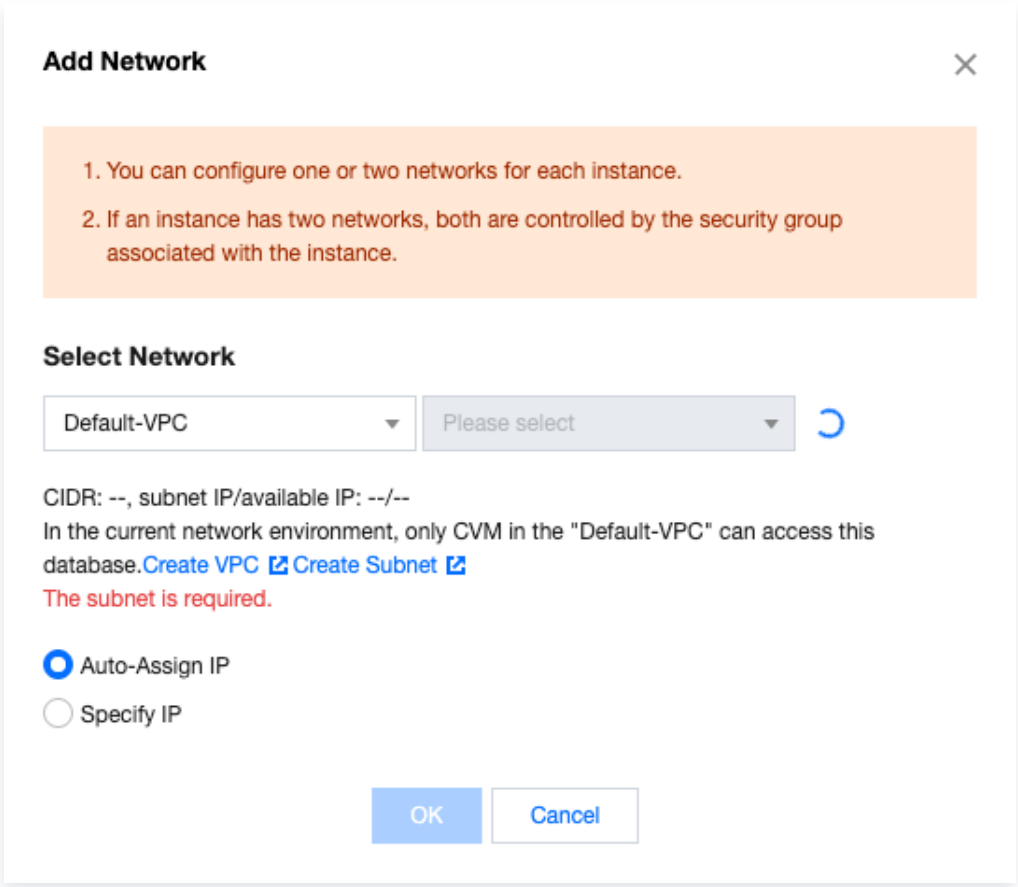
Add Network

1. Log in to the TencentDB for PostgreSQL console. In the instance list, click an instance ID or **Manage** in the **Operation** column to enter the instance management page.
2. In the instance details page, under **Basic Information** > **Network**, click **Add Network**.



3. In the pop-up dialog, select the network, set the IP automatically or specify the IP, and click **OK** after confirmation.

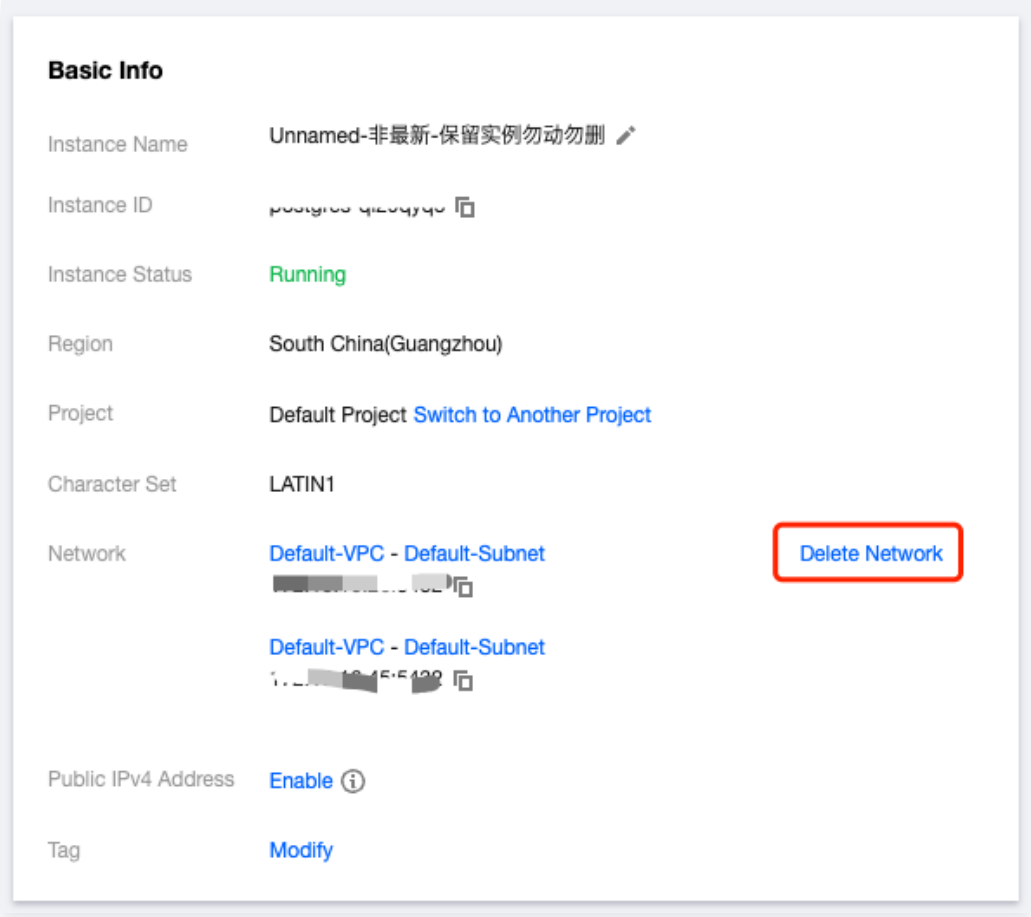
> **Note**
> - You can configure one or two networks for each instance.
> - If an instance has two networks, both are controlled by the security group associated with the instance.
> - You can only select VPC networks and subnets within the same region as the instance.



4. When the instance status changes from **Network Change in Progress** to **Running**, you can view the updated network information on the instance details page.

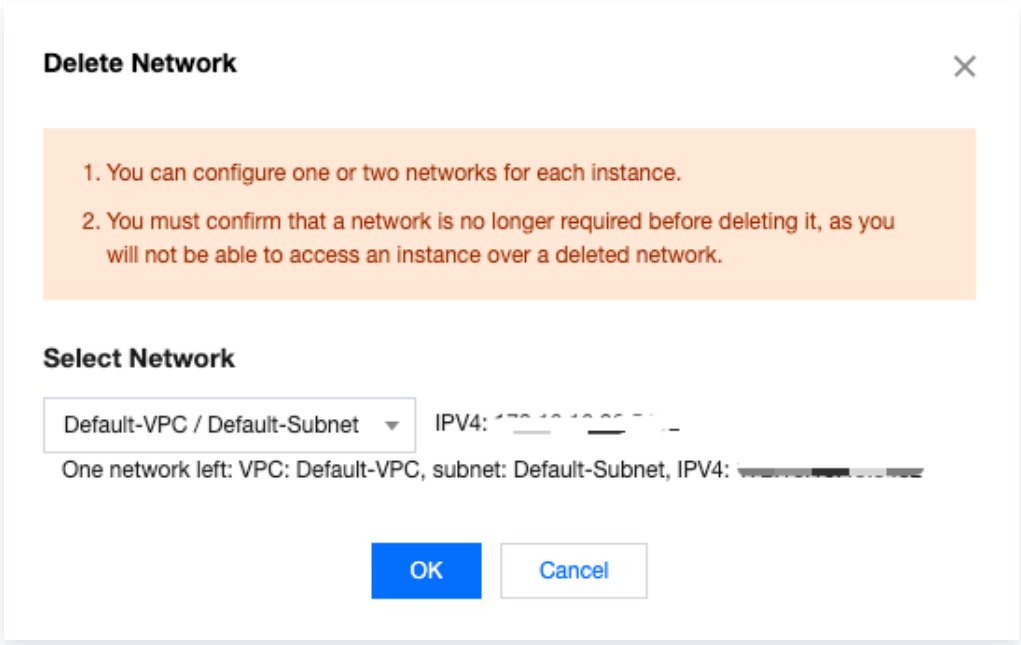
Delete Network

1. Log in to the TencentDB for PostgreSQL console. In the instance list, click an instance ID or **Manage** in the **Operation** column to enter the instance management page.
2. On the instance details page, click **Delete Network** next to **Network** in the **Basic Information** section.



3. In the pop-up dialog box, select the network to be deleted and click **OK**.

> **Note**
> - You can configure one or two networks for each instance.
> - You must confirm that a network is no longer required before deleting it, as you will not be able to access an instance over a deleted network.



4. When the instance status changes from **Network Change in Progress** to **Running**, you can view the updated network information on the instance details page.

Change Network

If you want to change the current network of the instance, for example, from the classic network to a VPC or from VPC A to VPC B, you can add and delete a network as detailed above for this need.

Example 1: changing from classic network to VPC

1. Log in to the TencentDB for PostgreSQL console. In the instance list, click an instance ID or **Manage** in the **Operation** column to enter the instance management page.
2. In the instance details page, go to **Basic Information** > **Network**, click **Add Network**, select the desired VPC, and click **Confirm**.
3. When the instance status is **Running**, click **Delete Network** next to the network, select the classic network, and click **Delete**. The instance's network will be changed from the original classic network to the new VPC.

Example 2: changing from VPC A to VPC B

1. Log in to the TencentDB for PostgreSQL console. In the instance list, click an instance ID or **Manage** in the **Operation** column to enter the instance management page.
2. In the instance details page, go to **Basic Information** > **Network**, click **Add Network**, select the desired VPC B, and click **Confirm**.
3. When the instance status is **Running**, click **Delete Network** next to the network, select VPC A, and click **Delete**. The instance's network will be changed from VPC A to VPC B.

> ⚠ **Note**

> After a network is deleted, you cannot access the instance over it. Make sure that a network is no longer needed before deleting it.

Cloud Access Management
CAM Overview

Last updated: 2023-09-10 09:10:34

Known Issues

If you have multiple users managing different Tencent Cloud services such as CVM, VPC, TencentDB for PostgreSQL, and other TencentDB products, and they all share your Tencent Cloud account access key, you may face the following problems:

- Your key will be easily compromised because it is shared by several users.
- Your users might introduce security risks from maloperations due to the lack of user access control.

Solution

You can mitigate these issues by using sub-accounts to assign different services to different users. By default, sub-accounts do not have permissions to use TencentDB for PostgreSQL or access its related resources. Therefore, you need to create policies that grant sub-accounts the necessary permissions to use the resources they require.

Cloud Access Management (CAM) is a Tencent Cloud service that helps you securely manage and control access permissions to your Tencent Cloud resources. Using CAM, you can create, manage, and terminate users (groups), and control the Tencent Cloud resources that can be used by the specified user through identity and policy management.

When using CAM, you can associate a policy with a user or user group to allow or forbid them to use specified resources to complete specified tasks. For more information on CAM policies, see Syntax Logic.

If you do not require access management for PostgreSQL-related resources for sub-accounts, you can skip this section. Skipping these parts will not affect your understanding and usage of the remaining portions of the documentation.

Getting Started

A CAM policy must either grant or deny the use of one or more PostgreSQL instances. Additionally, it must specify the resources that can be used for the operations (either all resources or a subset of resources for certain operations). The policy can also include conditions set for operating the resources.

Some PostgreSQL APIs do not support resource-level permissions, which means that you cannot specify resources when using those APIs.

Task scheduling	Document
Basic policy structure	Policy Syntax

Operation definition in a policy	PostgreSQL Operations
Resource definition in a policy	Resource Path for TencentDB for PostgreSQL
Resource-level Permissions Supported by PostgreSQL	Resource-level Permissions Supported by PostgreSQL
Console Samples	Console Samples

Authorization Policy Syntax

Last updated: 2023-09-10 09:10:44

Policy Syntax

CAM policy:

```
{
      "version":"2.0",
      "statement":
      [
        {
          "effect":"effect",
          "action":["action"],
          "resource":["resource"],
           "condition": {"key":{"value"}}
        }
      ]
}
```

- **version** is required. Currently, only the value "2.0" is allowed.
- **statement** describes the details of one or more permissions. It contains a permission or permission set of multiple other elements such as `effect`, `action`, `resource`, and `condition`. One policy has only one `statement`.
- **effect** is required. It describes the result of a statement. The result can be "allow" or an "explicit deny".
- **action** is required. It describes the allowed or denied action (operation). An operation can be an API or a feature set (a set of specific APIs prefixed with "permid").
- **resource** describes the objects the statement covers. A resource is described in a six-segment format. Detailed resource definitions vary by product. This element is required.
- **condition** describes the constraints for the policy to take effect. Conditions consist of operators, keys, and values. PostgreSQL currently does not support special conditions, so this item can be left unconfigured.

PostgreSQL Operations

In a CAM policy statement, you can specify any API operation from any service that supports CAM. For PostgreSQL, use APIs with the `postgres:` prefix, such as `postgres:DescribeDBInstances` or `postgres:DescribeDBInstanceAttribute`. To specify multiple operations in a single statement, separate them with commas as shown below:

```
"action":["postgres:action1","postgres:action2"]
```

You can also use wildcards to specify multiple actions. For example, you can specify all actions whose names begin with the word "Describe", as follows:

```
"action":["postgres:Describe*"]
```

To specify all PostgreSQL operations, use the wildcard (*) as shown below:

```
"action": ["postgres:*"]
```

PostgreSQL Resource Paths

Each CAM policy statement for PostgreSQL is resource-specific.
Resource paths are generally in the following format:

```
qcs:project_id:service_type:region:account:resource
```

- **project_id** describes the project information. It is only for compatibility with early CAM logic and does not need to be filled in.
  service_type is the product abbreviation, "postgres".
  region refers to the region information, such as "ap-shanghai".
  account is the owner's main account information (i.e., the "Account ID" on the account information page), such as "164xxx472".
  resource describes the detailed resource information of each product, such as "DBInstanceId/postgres-0xssvm8e" or "DBInstanceId/*". The table below describes the resources that can be used by PostgreSQL and the corresponding resource description methods.

Resources	Resource Description Method in Authorization Policy
Instance	qcs::postgres:$region:$account:DBInstanceId/$DBInstanceId

For example, you can specify an instance (instance ID: postgres-0xssvm8e) in the statement as shown below:

```
"resource":[ "qcs::postgres:ap-shanghai:164xxx472:DBInstanceId/postgres-0xssvm8e"]
```

You can also use the wildcard (*) to specify all instances in the Shanghai region that belong to a specific account as shown below:

```
"resource":[ "qcs::postgres:ap-shanghai:164xxx472:DBInstanceId/*"]
```

If you want to specify all resources or if a specific API operation does not support resource-level permission, you can use the wildcard (*) in the `resource` element as shown below:

```
"resource": ["*"]
```

To specify multiple resources in a single statement, separate them with commas. In the following example, we specified two instances:

```
"resource":["qcs::postgres::164xxx472:DBInstanceId/postgres-0xf1f41e","qcs::postgres::164xxx
```

Authorizable Resource Type

Last updated: 2023-09-10 09:10:52

Resource-Level Permission Overview

Resource-level permission refers to the ability to specify which resources a user has the capability to perform operations on. PostgreSQL partially supports resource-level permissions, allowing users to execute operations or access specific resources. The types of resources that can be authorized for PostgreSQL in CAM (Cloud Access Management) are as follows:

ResourceType	Resource Description Method in Authorization Policy
Instance	qcs::postgres:$region:$account:DBInstanceId/$DBInstanceId qcs::postgres:$region:$account:DBInstanceId/*

PostgreSQL Instance-related introduces the currently supported PostgreSQL API operations with resource-level permissions, as well as the resources and condition keys supported by each operation. When setting the resource path, you need to replace variables such as `$region` and `$account` with your actual parameter information. You can also use the * wildcard in the path. For related operation examples, please refer to Console Examples.

> **Note**
> For a PostgreSQL API operation that does not support authorization at the resource level, you can still authorize a user to perform the operation. In this case, you must specify `*` as the resource element in the policy statement.

List of APIs Not Supporting Resource-Level Permissions

Configuring using API	API Overview
CreateDBInstances	Create Instance
Create ServerlessDB Instance	Create ServerlessDB Instance
DescribeOrders	Getting order information
DescribeRegions	Queries purchasable regions
DescribeZones	Queries purchasable AZs
DescribeProductConfig	Queries purchasable specification configuration
InquiryPriceCreateDBInstances	Queries purchase price
DescribeServerlessDBInstances	Query ServerlessDB Instance List

List of APIs Supporting Resource-Level Permissions

PostgreSQL instance APIs

ServerlessDB Related Interfaces

API Name	Feature
CloseServerlessDBExtranetAccess	Disable public network access for ServerlessDB instance
DeleteServerlessDBInstance	Deleting ServerlessDB Instance
OpenServerlessDBExtranetAccess	Enable Public Network Access for ServerlessDB Instance

Backup and Restoration Related Interfaces

API Name	Feature
DescribeDBBackups	Query the instance backup list
DescribeDBErrlogs	Getting error logs
DescribeDBSlowlogs	Retrieve Slow Query Logs
DescribeDBXlogs	Retrieve Instance Xlog List

Instance-related APIs

API Name	Feature
CloseDBExtranetAccess	Disabling public network access for instance
DescribeDBInstanceAttribute	Queries instance details
DescribeDatabases	Pulling the list of databases
DestroyDBInstance	Terminating Instances
InitDBInstances	This example shows you how to initialize an instance.
InquiryPriceRenewDBInstance	Queries the renewal price of instance
InquiryPriceUpgradeDBInstance	Queries the upgrade price of instance
ModifyDBInstanceName	Modifies instance name
ModifyDBInstancesProject	Transferring instance to another project

OpenDBExtranetAccess	Enabling public network
RenewInstance	Renews instance
RestartDBInstance	Restart an instance
SetAutoRenewFlag	Auto-Renewal
UpgradeDBInstance	Upgrading Instance
DescribeDBInstances	Querying instance list

Account-related APIs

API Name	Feature
DescribeAccounts	Gets instance user list
ModifyAccountRemark	Modifying account remarks
ResetAccountPassword	Resetting Account Password

Console Examples

Last updated: 2023-09-10 09:11:02

Scenario

You can grant a user the permission to view and use specific resources in the PostgreSQL Console by using a CAM policy. This document provides examples of granting permissions to view and use specified resources, guiding users on how to apply specific policies within the console.

Instructions

Note

- To grant a user only the permissions of specific APIs, at least the permissions of the following APIs must be granted, or else the console fails to display correctly.
  The sample code of `action` is as follows:

```
"action": [
"postgres:DescribeProductConfig",
"postgres:InquiryPriceCreateDBInstances",
"postgres:DescribeRegions",
"postgres:DescribeZones"
]
```

- To grant a user the permissions to monitor and view instances, the API permissions related to monitoring needs to be granted. The sample code of `action` is as follows:

```
{"effect": "allow",
"action": [
"monitor:Get*",
"monitor:Describe*"
],
"resource": "*"
}
```

Full read/write permission policy for PostgreSQL

If you want a user to have the permission to create and manage PostgreSQL instances, you can apply the `QcloudPostgreSQLFullAccess` policy to that user.
This policy grants the user the ability to operate all resources within PostgreSQL. The specific steps are as follows:
Refer to Authorization Management and associate the preset policy `QcloudPostgreSQLFullAccess` with the user.

Read-only permission policy for PostgreSQL

If you want a user to have the permission to query PostgreSQL instances but not create, delete, or modify them, you can apply the `QcloudPostgreSQLReadOnlyAccess` policy to that user.
This policy grants the user permission to perform all operations in PostgreSQL that begin with the words "Describe" and "Inquiry". Follow these steps:
Refer to Authorization Management to grant the preset PostgreSQL policy to the user.

Policy for granting a user permissions to operate specific PostgreSQL instances

To grant a user the permission to manipulate a specific PostgreSQL cluster, you can associate the following policy with the user. The steps are as follows:

1. Create a custom policy according to Policy.
   This example policy grants the user all operation permissions for the PostgreSQL instance with the ID postgres-0xxxx8e. The policy content can be set by referring to the following policy syntax:

```
{
 "version": "2.0",
 "statement": [
    {
       "action": "postgres:*",
       "resource": "qcs::postgres:ap-shanghai:103xxx1481:DBInstanceId/postgres-0xxxx8e",
       "effect": "allow"
    }
 ]
}
```

2. Locate the created policy and click **Associate User/Group** in the **Operation** column.
3. In the "Associate User/User Group" window that pops up, select the user/group you want to authorize and click **Confirm**.

Policy for granting a user permissions to use all PostgreSQL resources

To grant a user the permission to manipulate all PostgreSQL resources, associate the following policy with the user. The steps are as follows:

1. Create a custom policy as instructed in Policy.
   This policy grants the user the permission to manipulate all PostgreSQL resources. The policy content can be set by referring to the following policy syntax:

```
{
 "version": "2.0",
 "statement": [
    {
       "action": "postgres:*",
       "resource": "qcs::postgres:::*",
```

```
      "effect": "allow"
    }
 ]
}
```

2. Locate the created policy and click **Associate User/Group** in the **Operation** column.
3. In the "Associate User/User Group" window that pops up, select the user/group you want to authorize and click **Confirm**.

Policy for denying a user permissions to operate specific PostgreSQL instances

To deny a user the permission to manipulate certain PostgreSQL instances, associate the following policy with the user. The steps are as follows:

1. Create a custom policy as instructed in Policy.
   This example policy prohibits the user from having operational permissions on instances with IDs postgres-c8xxxa4 and postgres-d8xxxb4. The policy content can be set by referring to the following policy syntax:

```
{
 "version": "2.0",
 "statement": [
    {
      "action": "postgres:*",
      "resource": [
              "qcs::postgres::16xxx472:DBInstanceId/postgres-c8xxxa4",
              "qcs::postgres::16xxx472:DBInstanceId/postgres-d8xxxb4",
              ],
      "effect": "deny"
    }
 ]
}
```

2. Locate the created policy and click **Associate User/Group** in the **Operation** column.
3. In the "Associate User/User Group" window that pops up, select the user/group you want to authorize and click **Confirm**.

Custom Policy

If you find that the preset policies do not meet your requirements, you can achieve your goal by creating a custom policy.
For detailed instructions, please refer to Create a Custom Policy.
For more PostgreSQL-related policy syntax, please refer to Authorization Policy Syntax.

Security Groups
Managing Security Group

Last updated: 2023-09-10 09:12:08

Scenario

Security group is a stateful virtual firewall with filtering capabilities, used to set network access control for single or multiple cloud databases. It is an important means of network security isolation provided by Tencent Cloud. A security group is a logical grouping, allowing you to add cloud database instances with the same network security isolation requirements within the same region to the same security group. Cloud databases and CVMs share the security group list, with rules based on matching within the group. For specific rules and limitations, please refer to Security Group Detailed Description.

Note

- TencentDB for PostgreSQL security groups currently only support network access control for VPCs and public networks but not the classic network.
- Security groups that currently support public network access are available only in the Beijing, Shanghai, Guangzhou, and Chengdu regions.
- As TDSQL-A for PostgreSQL does not have active outbound traffic, outbound rules are not applicable to it.
- TencentDB for PostgreSQL primary instances, read-only instances, and read-only instance groups (RO groups) support security groups.

Configuring a Security Group

Step 1. Create a security group

1. Log in to the CVM console.
2. Select **Security Group** on the left sidebar, choose a region, and click **Create**.
3. In the pop-up window, set the following configuration items, confirm that everything is correct, and click **OK**.

- **Template**: Select a template based on the service to be deployed on the TencentDB instance in the security group, which simplifies the security group rule configuration, as shown in the table below.

Template	Note	Scenes
Open all ports	Opening all ports to both public and private networks by default poses certain security risks.	-

Allow ports 22, 80, 443, 3389 and the ICMP protocol.	By default, open ports 22, 80, 443, 3389, and the ICMP protocol, and allow all traffic within the private network.	This template does not apply to cloud databases.
Custom	After successfully creating a security group, add security group rules as needed. For detailed instructions, please refer to the "Adding Security Group Rules" section below.	It is recommended to choose this template and customize one for accessing specific instances.

- **Name**: Custom name of the security group.
- **Project**: Select a project for easier management. By default, **DEFAULT PROJECT** is selected.
- **Notes**: A short description of the security group for easier management.

Step 2: Add Security Group Rules

1. On the Security Group page, locate the security group you want to set rules for, and click **Modify Rules** in the **Actions** column.
2. On the Security Group Rules page, select **Inbound Rules** > **Add Rule**.
3. In the pop-up dialog box, set the rule.

- **Type**: **Custom** by default.
- **Source** or **Destination**: traffic origin (inbound rules) or target (outbound rules). You can use one of the following to define Source or Destination:

Specified Source/Destination	Note
A single IPv4 address or an IPv4 address range	A single IPv4 address or an IPv4 range is represented in CIDR notation, such as 203.0.113.0, 203.0.113.0/24, or 0.0.0.0/0, where 0.0.0.0/0 indicates all IPv4 addresses will be matched.
A single IPv6 address or an IPv6 address range	CIDR notation is used for IPv6 addresses, such as FF05::B5, FF05:B5::/60, ::/0, or 0::0/0, where ::/0 or 0::0/0 represents matching all IPv6 addresses.
To reference a security group ID, you can use the ID of the following security groups: Security Group ID Other Security Groups	The current security group represents the CVMs associated with the security group. Other security groups refer to another security group ID within the same region and project.
Refer to the IP address objects or IP address group objects in the Parameter Template.	-

- Protocol port: Enter the protocol type and port range, or you can reference the protocol port or protocol port group from the Parameter Template.

> **Note**
> To connect to TencentDB for PostgreSQL, port 5432 must be opened.

- Policy: Allow or Refuse. By default, Allow is selected.
  - Allow: traffic to this port is allowed.
  - Reject: Discard all data packets going to this port without any response.
- Remarks: a short description of the security group rule.

4. Click **Complete** to finish adding the inbound security group rule.

Case Study

Scenario: You have created a TencentDB for PostgreSQL instance and want to access it through a CVM instance.
Solution: When adding security group rules, configure access permissions for the TCP:5432 protocol port in the inbound rules.
You can also open access to all IPs or specific IPs (IP ranges) based on your needs, and configure the IP sources that can access the TencentDB for PostgreSQL instance through the CVM.

Direction	Local Disk Types	Source	Protocol port	Rule
Inbound direction	Custom	All IPs: 0.0.0.0/0 Designated IP: Enter the specified IP or IP range	TCP:5432	Supported

Import Security Group Rules

1. On the Security Group page, select the desired security group and click on the security group ID/name.
2. On the Inbound/Outbound Rules tab, click **Import Rules**.
3. In the pop-up window, select the edited inbound/outbound rule template file and click **Start Import**.

> **Note**
> If the security group you wish to import rules into already has existing rules, it is recommended to export the current rules first; otherwise, importing new rules will overwrite the existing ones.

Clone Security Group

1. On the Security Groups page, in the **Operation** column, select **More** > **Clone**.

2. In the pop-up dialog, select the target region and target project, then click **OK**. If the new security group needs to be associated with a CVM, please manage the CVM within the security group again.

Deleting Security Group

1. On the Security Group page, select the security group you want to delete, and in the **Action** column, choose **More** > **Delete**.
2. In the pop-up dialog box, click **OK**. If the current security group is associated with a CVM, you must first unbind the security group before deleting it.

Associating Instance with Security Group

Last updated: 2023-09-10 09:12:16

A security group is an instance-level firewall provided by Tencent Cloud for controlling inbound traffic of TencentDB. You can associate a security group with an instance when purchasing it or later in the console.
In TencentDB for PostgreSQL, primary instances, read-only instances, and read-only instance groups (RO groups) can use security groups which are independent from each other.

Note

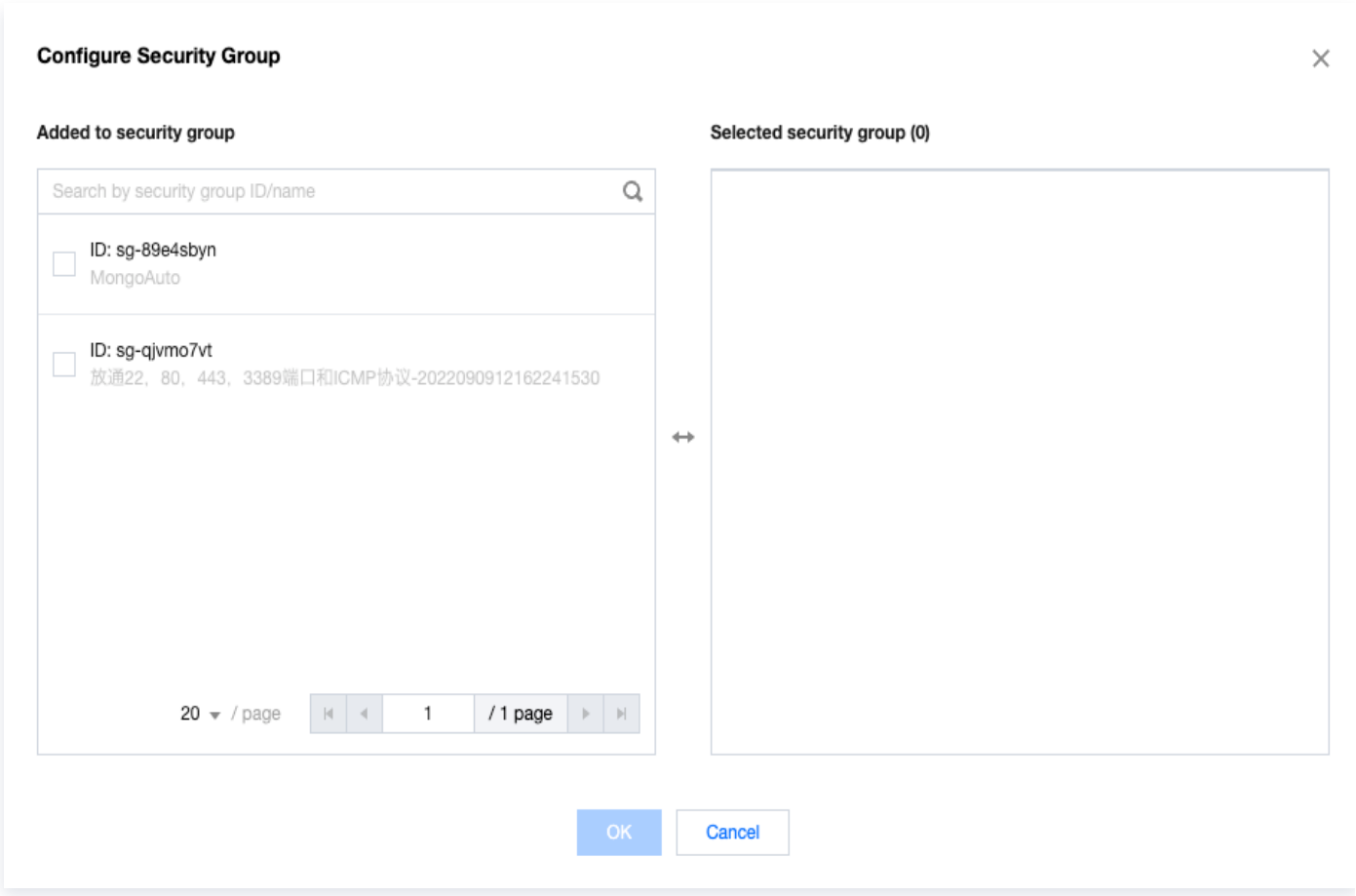
- A TencentDB for PostgreSQL instance can associate with up to five security groups.
- Associating a security group with a read-only instance group does not affect the connections to the read-only instance's access address, but only the connections to the instance through the read-only instance group's access address. For example, if the security group for a read-only instance denies access from a specific IP, but the security group for the read-only instance group allows access from that IP, the user can access the read-only instance through the read-only instance group's access address. However, access through the read-only instance's own access address will be denied.

Preparations

Refer to Managing Security Groups to create a security group for the instance in the security group console.

Associating Security Groups with Primary/Read-Only Instances

1. Log in to the TencentDB for PostgreSQL Console. In the instance list, click the instance ID of the instance you want to associate with a security group, and enter the instance management page.
2. On the **Security Group** tab, click **Configure Security Group**.
3. In the pop-up dialog, select the security group to be associated, and click **OK** to complete the process of binding the security group to the TencentDB instance.



Associating Security Groups with RO Groups

1. Log in to the TencentDB for PostgreSQL console. In the instance list, locate the read-only instance managed by the read-only instance group, click the instance ID, and enter the read-only instance management page.
2. On the **Security Group** tab of the read-only instance management page, select the object type as "Read-Only Instance Group" in the "Security Group Effective Objects" section, and then click **Configure Security Group**.
3. In the pop-up dialog, select the security group to be associated, and click **OK** to complete the process of binding the security group to the read-only instance group.

Adjusting Security Group Priority

You can associate up to five security groups with a TencentDB instance. If you have associated multiple security groups, these security groups are executed based on their priorities. You can adjust the priorities as follows.

1. Log in to the TencentDB for PostgreSQL console. In the instance list, click the instance ID to enter the instance management page.
2. On the **Security Group** tab in the Instance Management page.

3. In the "Joined Security Group" section, click **Edit**, select the up and down buttons on the right side of the security group to adjust its priority. The higher the position, the higher the priority of the security group. All security group configurations are combined. If two conflicting security group configurations are set, the one with the higher priority will prevail.
4. After making adjustments, click **Save**.

Monitoring and Alarming Monitoring

Last updated: 2023-09-10 09:12:25

To facilitate users in viewing and understanding the operational information of their instances, TencentDB for PostgreSQL offers a wide range of performance monitoring metrics. Users can log in to the PostgreSQL Console and view these metrics on the **System Monitoring** page of the corresponding instance management page.

Metric Name	Monitoring Metric Name	<p>	Metric Description
CPU utilization	cpu	%	Actual CPU Utilization.
Used Storage Space	storage	GB	Total space occupied by the instance.
Data file size	data_file_size	GB	Data file occupied space size.
Log file size	log_file_size	GB	Size of the occupied space by WAL log files.
Temp File Size	temp_file_size	MB	Size of temporary files.
Storage space utilization	storage_rate	%	Total storage usage rate, including temporary files, data files, log files, and other types of database files.
Queries per second	qps	Queries/second	Average number of SQL statements executed per second.
Number of connections	connections	Connections	The total number of connections to the database when data collection is initiated.
Connections Created in the Last 5 Sec	new_conn_in5s	Threads	When initiating data collection, the number of all connections established within the last 5 seconds is queried.
Number of active connections	active_conns	Connections	Number of instantaneous active connections (non-idle connections) to the database when data collection is initiated.
Idle Connections	idle_conns	Connectio	The instantaneous idle

		ns	connections (idle connections) of the database when initiating data collection through a query.
Waiting Sessions	waiting	Connections	The number of sessions waiting (in the waiting state) when data collection is initiated on the database.
Sessions Waiting for More Than 5 Sec	long_waiting	Connections	The number of sessions in a waiting state and exceeding 5 seconds when initiating data collection on the database (status: waiting).
Idle Transactions	idle_in_xact	Connections	The number of transactions in the 'idle in transaction' state when data collection is initiated for the database.
Number of transactions with an execution duration exceeding 1 second	long_xact	Connections	Number of transactions with an execution time exceeding 1 second when initiating data collection on the database.
Transactions Idle for More than 5 Sec	long_idle_in_xact	Connections	The number of transactions in the "idle in transaction" state and lasting more than 5 seconds when initiating collection on the database.
Transactions per Second	tps	Queries/second	Average number of successful transactions executed per second (including rollbacks and commits).
Transactions Committed /sec	xact_commit	Queries/second	Average number of transactions committed per second within a statistical period.
Rolled-Back Transactions	xact_rollback	Queries/second	Average number of transactions rolled back per second within a statistical period.
Read-Write Requests	read_write_calls	Threads	Number of read/write requests within a statistical period.
Read Requests	read_calls	Threads	Number of read requests within a statistical period.
Number of write	write_calls	Threads	Number of write requests within a

requests			statistical period.
Other Requests	other_calls	Threads	Number of other requests (excluding SELECT, INSERT, UPDATE, and DELETE) within a statistical period.
Buffer Cache Hit Ratio	hit_percent	%	Hit rate of all SQL statement executions within a statistical period.
Average execution latency	sql_runtime_avg	ms	The average execution latency of all SQL statements within a single statistical period.
Top 10 Longest Execution Latency	sql_runtime_max	ms	Average execution latency of the top 10 longest SQL queries within a single statistical period.
Top 10 Shortest Execution Latency	sql_runtime_min	ms	Average of the Top 10 Shortest SQL Execution Latencies within a single statistical period.
Remaining XIDs	remain_xid	Connections	When initiating data collection for a database, the remaining XID count of the database with the least remaining XIDs is displayed. This metric is not applicable to read-only instances.
Difference in standby log send and replay positions (bytes)	xlog_diff	byte	The size difference between the log sent from the primary database to the replica database and the completion of the replica's log playback primarily reflects the speed of log application on the replica. This metric mainly allows for the assessment of the replica's performance and the speed of network transmission. Read-only instances do not have this metric.
Standby WAL Flush Lag (s)	xlog_diff_time	byte	The time difference between when logs are sent from the primary instance to the replica instance and when the replica instance receives and writes the logs to disk. This metric is not available for read-only instances

			and is only available for instances with version 10.x or higher.
Master/Replica Data Synchronization Delay (Bytes)	slave_apply_delay	s	The difference in size between the replica's replay LSN and the primary instance's current LSN. For the primary instance, this metric reflects the Recovery Time Objective (RTO) during a failover. For read-only instances, this metric indicates the extent of data latency.
Slow Query Count	slow_query_cnt	Connections	The number of slow queries occurring within a single collection period.
Number of SQL statements with execution duration exceeding 1 second	long_query	Connections	The number of SQL statements being executed and having an execution time exceeding 1 second when the database is being collected.
Two-Phase Commit Transactions	2pc	Connections	The number of 2PC (two-phase commit) transactions when initiating data collection on the database.
Number of 2PC Transactions Uncommitted for Over 5 Seconds	long_2pc	Connections	The number of 2PC transactions with a current execution time exceeding 5 seconds when initiating data collection on the database.
Rows Deleted /sec	tup_deleted	Connections	Average number of tuples deleted per second within a statistical period. This metric is not applicable to read-only instances.
Insertions per second	tup_inserted	Connections	Average number of tuples inserted per second within a statistical period. This metric is not applicable to read-only instances.
Updates per second	tup_updated	Connections	Average number of tuples updated per second within a statistical period. This metric is

			not applicable to read-only instances.
Index Scans with Backward Record Fetches per Second	tup_fetched	Connections	Average number of tuples scanned per second using index within a statistical period.
Records Scanned per Second	tup_returned	Connections	Average number of tuples scanned per second in a full table scan during a statistical period.
Deadlocks	deadlocks	Connections	Total number of deadlocks within a statistical period.

Tag
Tag Overview

Last updated: 2023-09-10 09:13:22

Feature Overview

Tags are key-value pairs provided by Tencent Cloud for identifying cloud resources. For more information, see Tag Overview.
You can use tags to categorize and manage TencentDB for PostgreSQL resources based on various dimensions (such as business, purpose, and person in charge). Tags make it easy to filter and find corresponding resources. Tencent Cloud does not assign any semantic meaning to tag key-value pairs and strictly parses them as strings. Just be aware of the usage restrictions during use.
The following is a specific example to demonstrate the use of tags.

Case Background

A company has three PostgreSQL instances in Tencent Cloud. Those instances are distributed in three gaming businesses whose OPS owners are John, Jane, and Harry.

Configuring Tags

To manage the resources better, the company categorizes its TencentDB for PostgreSQL resources with tags and defines the following tag key-value pairs.

Tag Key	Tag Value
Business data collection	Game 1, Game 2, Game 3
Ops Owner	John, Jane, and Harry

These tag key-value pairs are bound to TencentDB for PostgreSQL instances in the following way:

instance-id	Business data collection	Ops Owner
postgres-abcdef1	Game 1	Chris
postgres-abcdef2	Game 2	Jane Smith
postgres-abcdef3	Game 3	John Smith

Using Tags

- For instructions on creating and deleting tags, see Tag Quick Start.
- For instructions on editing tags for TencentDB for PostgreSQL, see Editing Tags.