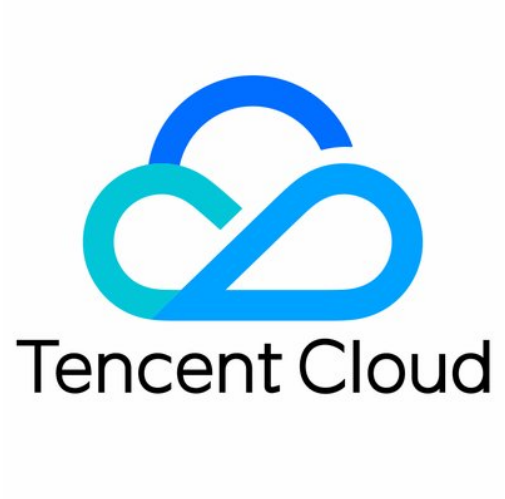
TencentDB for PostgreSQL

PostgreSQL for Serverless

Product Documentation



Contents

PostgreSQL for Serverless Overview

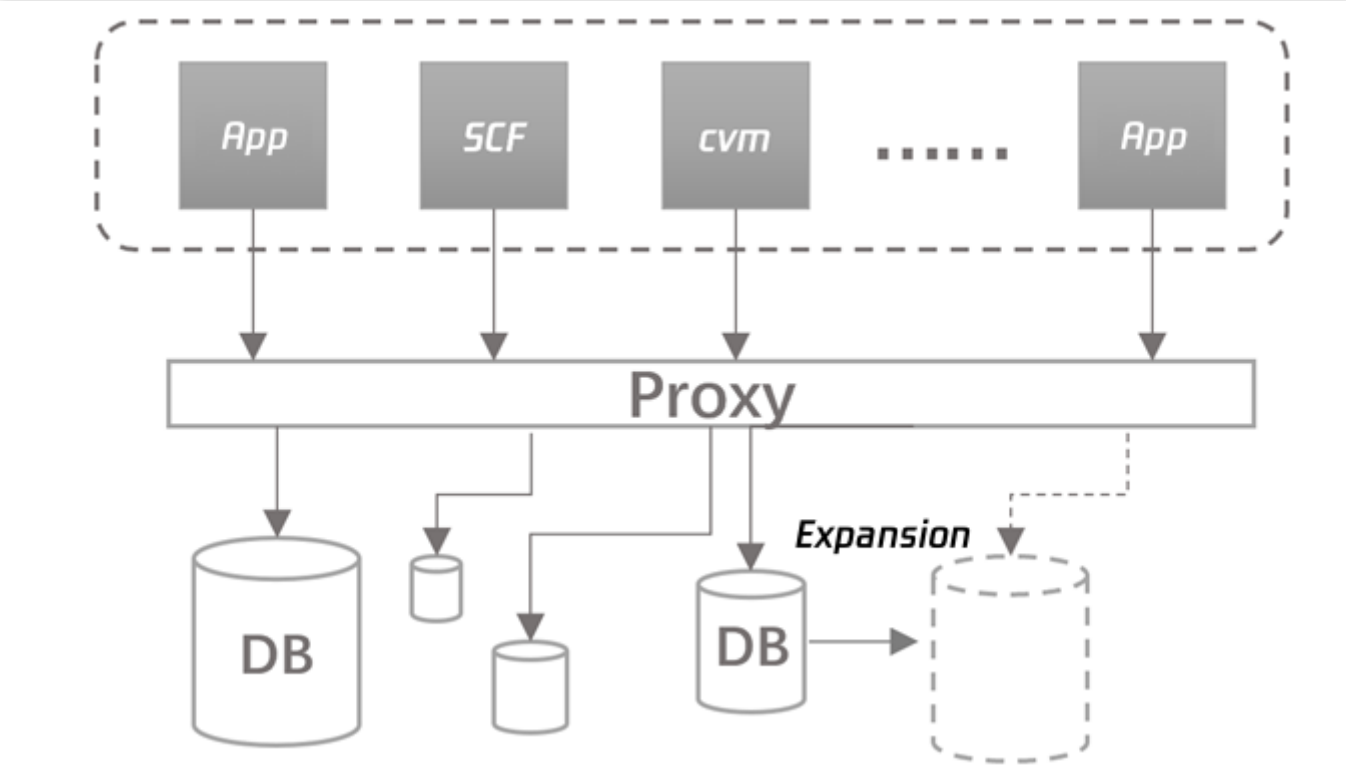
Last updated：2024-01-24 11:20:59

Overview

PostgreSQL for Serverless (ServerlessDB) is a PostgreSQL-based database product that enables on-demand allocation of resources. It can automatically allocate resources according to the actual number of requests.
With a traditional database instance, you need to manually adjust the database specification and capacity according to the actual business usage, which not only takes up your time but also may cause resource waste due to the underuse of database resources.
With PostgreSQL for Serverless, you can create a database instance for easy use without caring about the instance specifications. You only need to pay for the actual usage when the database is active.
Note:
PostgreSQL for Serverless is currently in beta test free of charge. You can create an instance directly through the CreateServerlessDBInstance API.

System Architecture

After users are connected to a database, requests will be forwarded uniformly through the PostgreSQL for Serverless proxy layer before data operations are performed. When the number of user requests increases, the database will automatically respond. Currently, up to 40,000 QPS is supported for one single user.



Features

High availability

PostgreSQL for Serverless supports the architecture of one primary instance and one secondary instance for high availability. When the primary instance becomes unavailable due to an exception, the database will automatically start the secondary instance, to which business connections will be transferred so as to prevent business interruption.

Automatic backup

PostgreSQL for Serverless automatically backs up the entire database at 1:00 AM every day and backs up database logs once every 15 minutes or when the number of log files reaches 60. All backups are retained for 7 days.
Backup is automatically performed on the backend. Currently, you cannot view, download, or restore data from backups.

Strengths

Last updated：2024-01-24 11:20:59

Automatic Database Capacity Management

PostgreSQL for Serverless can automatically respond to resources based on the busyness of business requests, so you don't need to pay for the database during idle time; instead, you only need to pay for the actually used data capacity and response resources when the database is active, which helps your save database costs. If there are no user requests to the database, the database will automatically close all resource responses.

High Availability

PostgreSQL for Serverless supports the architecture of one primary instance and one secondary instance for high availability. When the primary instance becomes unavailable due to an exception, the database will automatically start the secondary instance, to which business connections will be transferred so as to prevent business interruption.

Automatic Backup

PostgreSQL for Serverless automatically backs up the entire database at 1:00 AM every day and backs up database logs once every 15 minutes or when the number of log files reaches 60. All backups are retained for 7 days.
Backup is automatically performed on the backend. Currently, you cannot view, download, or restore data from backups.

Cost Savings

With PostgreSQL for Serverless, you can create a database instance for easy use without caring about the instance specifications. You only need to pay for the actual usage when the database is active.

Support for Features of PostgreSQL

In addition to supporting the serverless architecture, PostgreSQL for Serverless also enjoys the functional advantages of standard PostgreSQL database instances, such as rich extensions, backup and restoration, and high availability.

Use Cases

Last updated：2024-01-24 11:20:59

Infrequently Used Application

Some applications are used only a few times a day or a week for only a few minutes each time, such as a low-volume blog website. If you choose a traditional database, you need to pay for the database even when it is idle.
With PostgreSQL for Serverless, you can create a cost-effective database and only pay for it when it is active.

Unpredictable Workload

If your program needs to use the database around the clock and has unpredictable activity peaks, or when the program load changes rapidly, unpredictable business peaks may occur at any time, then PostgreSQL for Serverless is an ideal choice for you.
With PostgreSQL for Serverless, the capacity of your database will be automatically expanded to sustain peak loads of your application and reduced after activity surges end. There is no need to provision the peak capacity, so you don't have to pay for infrequently used resources, and there is no need to evenly allocate the capacity, so an optimal balance can be achieved between the performance and cost.

Development and Test Databases

Software development and quality assurance (QA) teams need to use databases during working hours but not at night or on weekends.
With PostgreSQL for Serverless, your database will be automatically shut down when not in use and started up faster when you start working in it the next day.

Low-Traffic Application

If your application has low traffic, it is usually impossible for you to make the most out of the standard instance with the lowest specification, but you still need to pay for the surplus performance.
PostgreSQL for Serverless eliminates your need to pay for the surplus performance.

Getting Started

Last updated：2024-01-24 11:20:59

This document describes how to create and connect to a PostgreSQL for Serverless instance.

Use Limits

During the beta period, a single user can enjoy up to 40,000 QPS and 100 GB disk capacity.
Currently, PostgreSQL for Serverless instances can be created only with APIs.
Currently, only Beijing Zone 3, Shanghai Zone 2, and Guangzhou Zone 2 are supported.

Directions

Step 1. Create an instance

Using Serverless Framework to create an instance

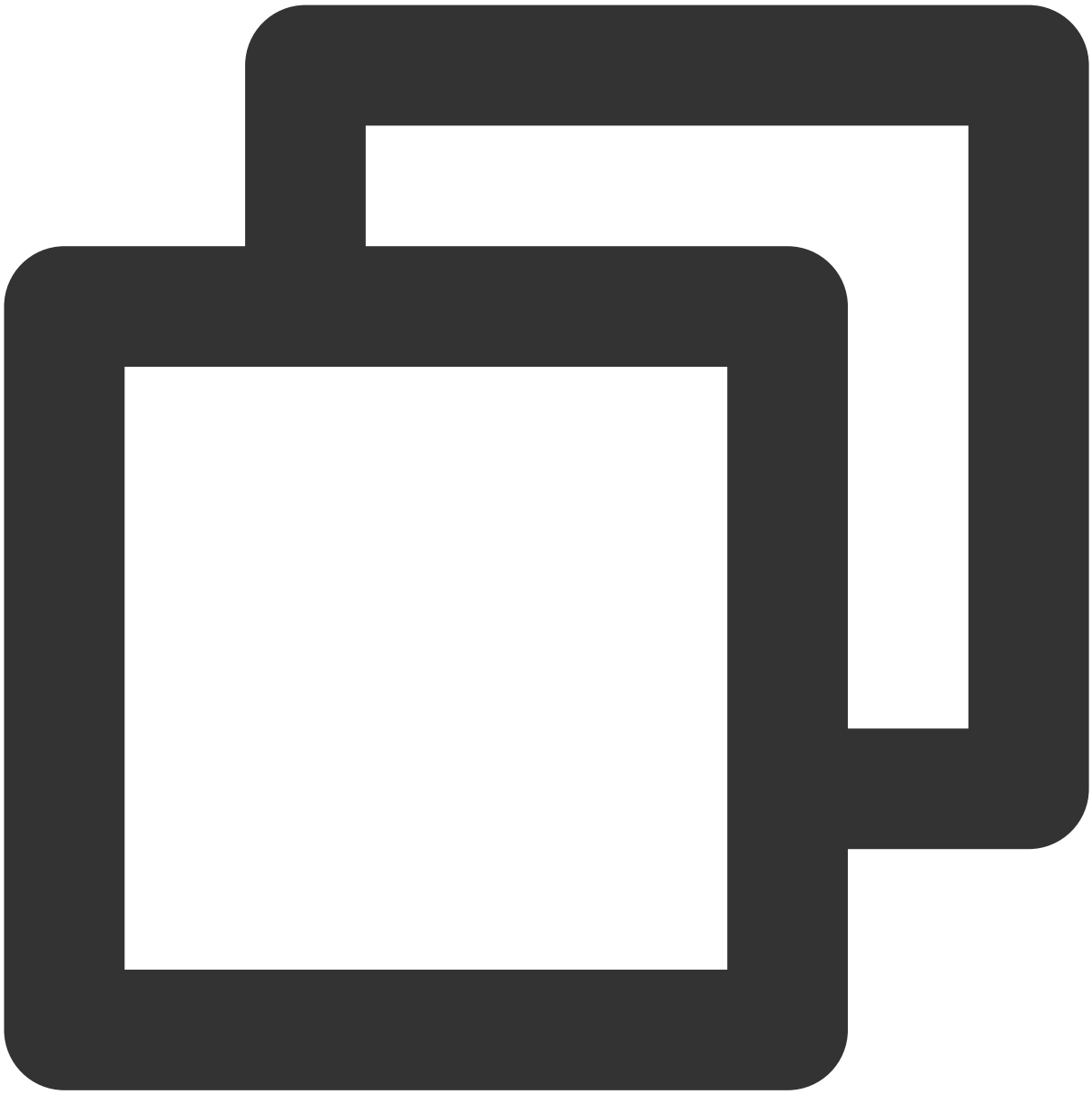
Create an instance quickly with Serverless Framework.

Using APIs to create an instance

Create an instance with the CreateServerlessDBInstance API.
Some important input parameters are listed as follows. For more information, please see CreateServerlessDBInstance.

Parameter Name	Required	Type	Description
Zone	Yes	string	Availability zone ID. Valid values: ap-shanghai-2, ap-beijing-3, ap-guangzhou-2
DBInstanceName	Yes	string	TencentDB instance name. This value must be unique for the same account.
DBVersion	Yes	string	Database version. Valid value: 10.4
DBCharset	Yes	string	PostgreSQL database character set. Valid values: UTF8, LATIN1
VpcId	No	string	VPC ID. If this parameter is left empty, the instance will be assigned with a classic network IP.
SubnetId	No	string	VPC subnet ID. This parameter should be used together with `VpcId` .

After successful execution, the output sample is as follows:

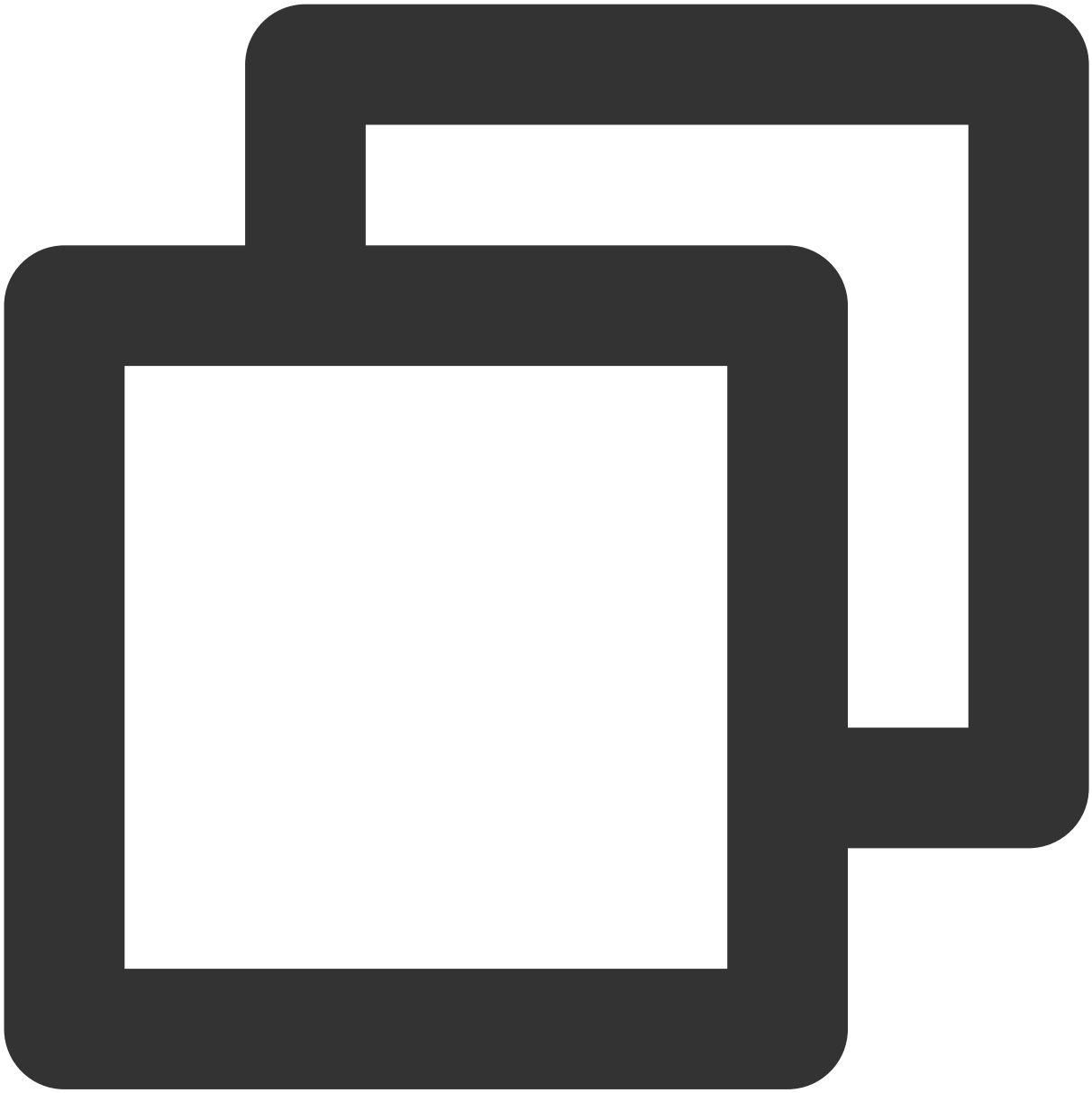


```
{
  "Response": {
    "RequestId": "20304c-6fd7-4427-8e09-2b081e1",
    "DBInstanceId": "postgres-xxxxxxx"
  }
}
```

Where, `DBInstanceId` refers to the instance ID.

Step 2. Connect to the instance

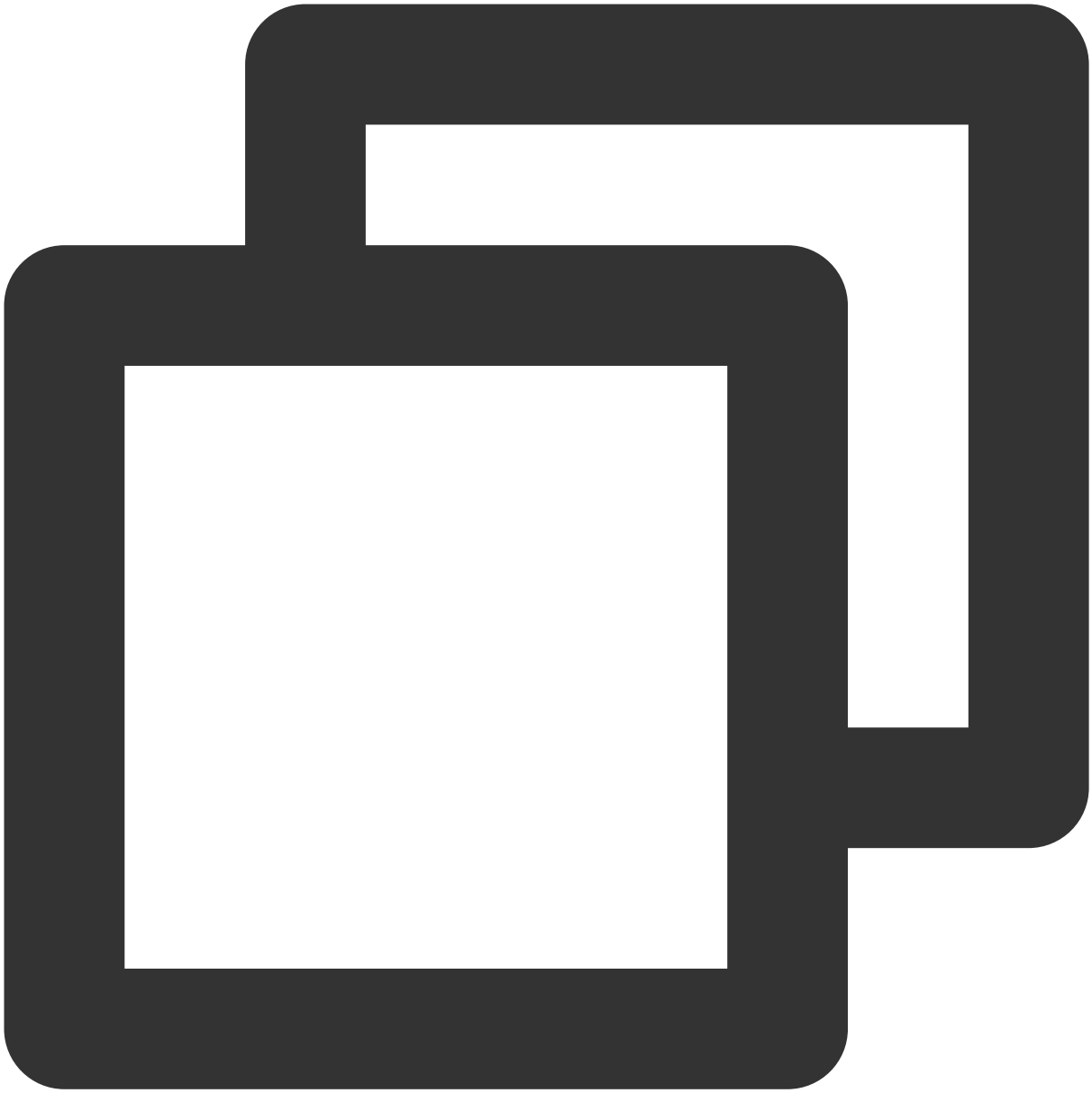
1. Use the DescribeServerlessDBInstances API to query the information of the PostgreSQL for Serverless instance you just created, including instance IP, port, username, and initial password.




```
{
  "Response": {
 "TotalCount": 1,
 "DBInstanceSet": [
   {
     "DBInstanceId": "postgres-xxxxxxx",
     "DBInstanceName": "test",
```

```
      "DBInstanceStatus": "running",#TencentDB instance status
      "Region": "ap-shanghai",
      "Zone": "ap-shanghai-2",
      "ProjectId": 0,
      "VpcId": "vpc-test",
      "SubnetId": "subnet-test",
      "DBCharset": "UTF8",
      "DBVersion": "10.4",
      "CreateTime": "2020-03-23 11:43:56",
      "DBInstanceNetInfo": [
        {
          "Address": "",
          "Ip": "10.1.1.2", #This IP is used as an example. The IP can be accessed o
          "Port": 5432, #Connect to the port
          "Status": "opened",
          "NetType": "private"
        },
        {
          "Address": "",
          "Ip": "",
          "Port": 0,
          "Status": "0",
          "NetType": "public"
        }
      ],
      "DBAccountSet": [
        {
          "DBUser": "tencentdb_xxxxxxx",
          "DBPassword": "***************",#Database password. After the password has
          "DBConnLimit": 100
        }
      ],
      "DBDatabaseList": [
        "tencentdb_xxxxxxx"
      ]
    }
  ],
  "RequestId": "89583d-cfdd-4db1-bd32-64eb1dbfa"
   }
}
```


2. Taking a CVM instance on CentOS 7.2 (64-bit) as an example, run the following command to install the PostgreSQL client.



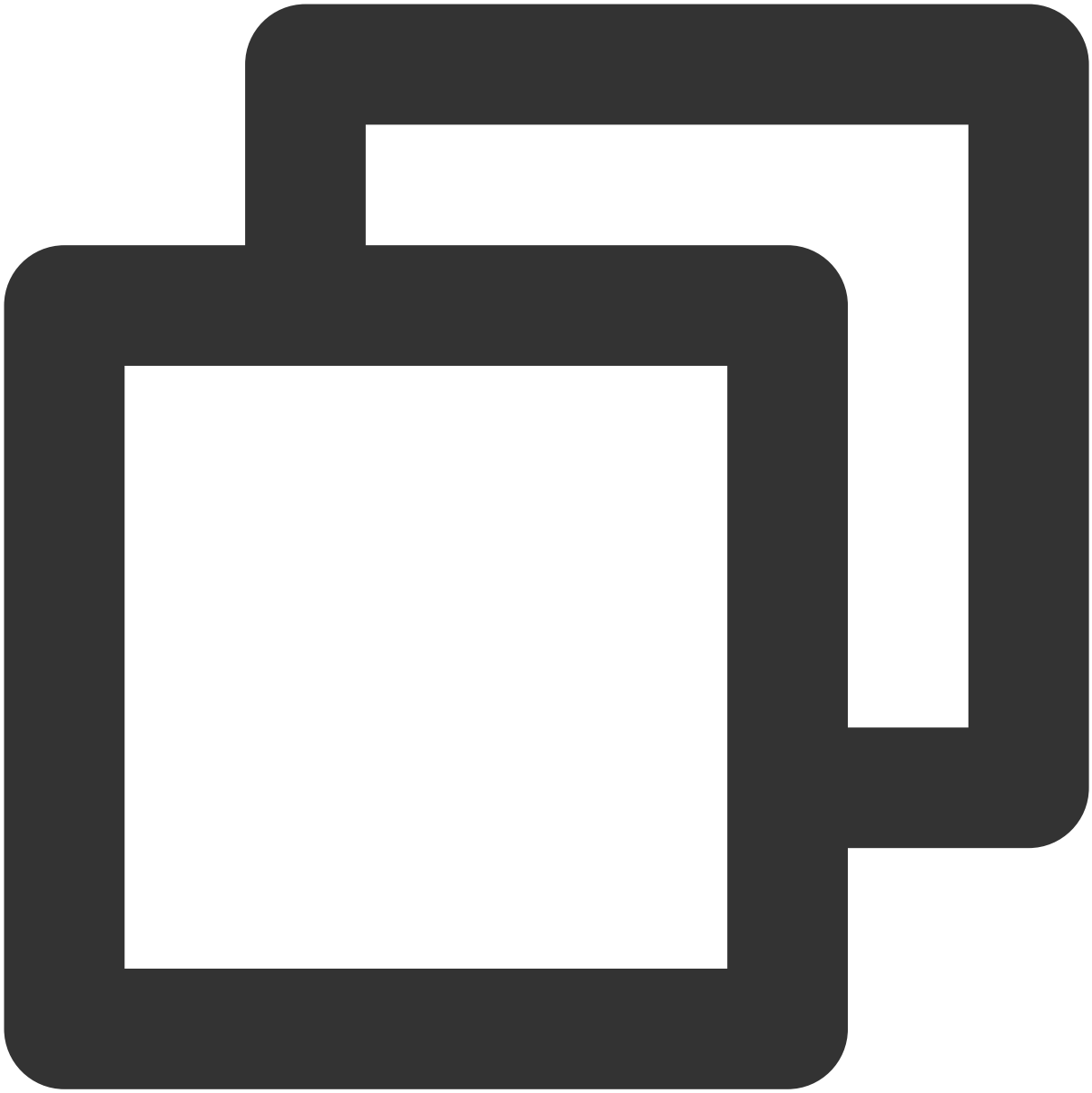
```
yum install https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg
yum install postgresql10 postgresql10.x86_64
```

3. Run the following command to connect to the database.

Note:

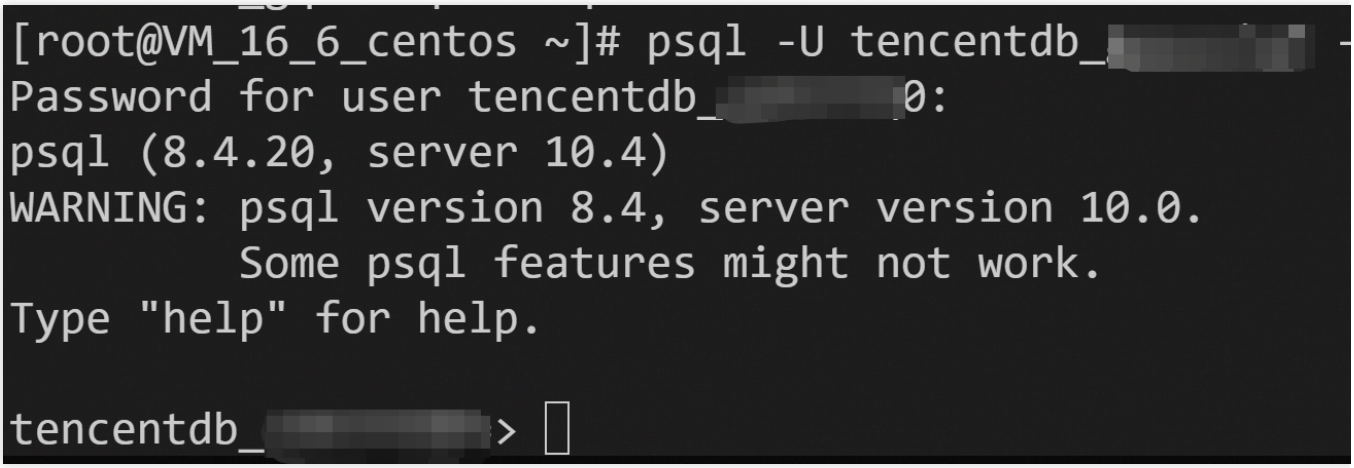
To connect over a private network, you need to use a CVM instance to access the private IP of the TencentDB instance, and the two instances must be under the same account and in the same VPC in the same region.

To connect over a public network, you need to enable public access for the instance.



```
psql -U database username -h IP address -p port number
```

In this example, the prompt `tencentdb_xx>` indicates successful connection.



4. After successful connection, you can manage database data with SQL statements. For more information, please see Official Document.

Note:

PostgreSQL for Serverless does not support the following operations:

Create a database

Access the `postgres` system database

View database parameters

SET/RESET statements

LOAD statements

PRESERVE/DELETE ROWS temp tables+

LISTEN/NOTIFY

WITH HOLD CURSOR

PREPARE/DEALLOCATE

Appendix. Enabling Public Network Access for Instances

To access an instance over a public network, you can use the OpenServerlessDBExtranetAccess API to enable the public network access for the instance.

Note:

After public network access is enabled, the database can be connected and accessed over a public network, which poses security risks. We recommend you access your database instances over a private network.

References

You can use the CloseServerlessDBExtranetAccess API to disable public network access for an instance.

You can use the DeleteServerlessDBInstance API to terminate an instance.

Note:

Please note that once an instance is terminated, its data cannot be restored.

If you want to import data to a PostgreSQL for Serverless instance, please see Importing Data.

You can use Serverless Framwork to quickly create PostgreSQL for Serverless-based web applications.

Importing Data

Last updated：2024-01-24 11:20:59

Currently, you can import data backed up from `pg_dump` to PostgreSQL for Serverless through the `psql` command.

Directions

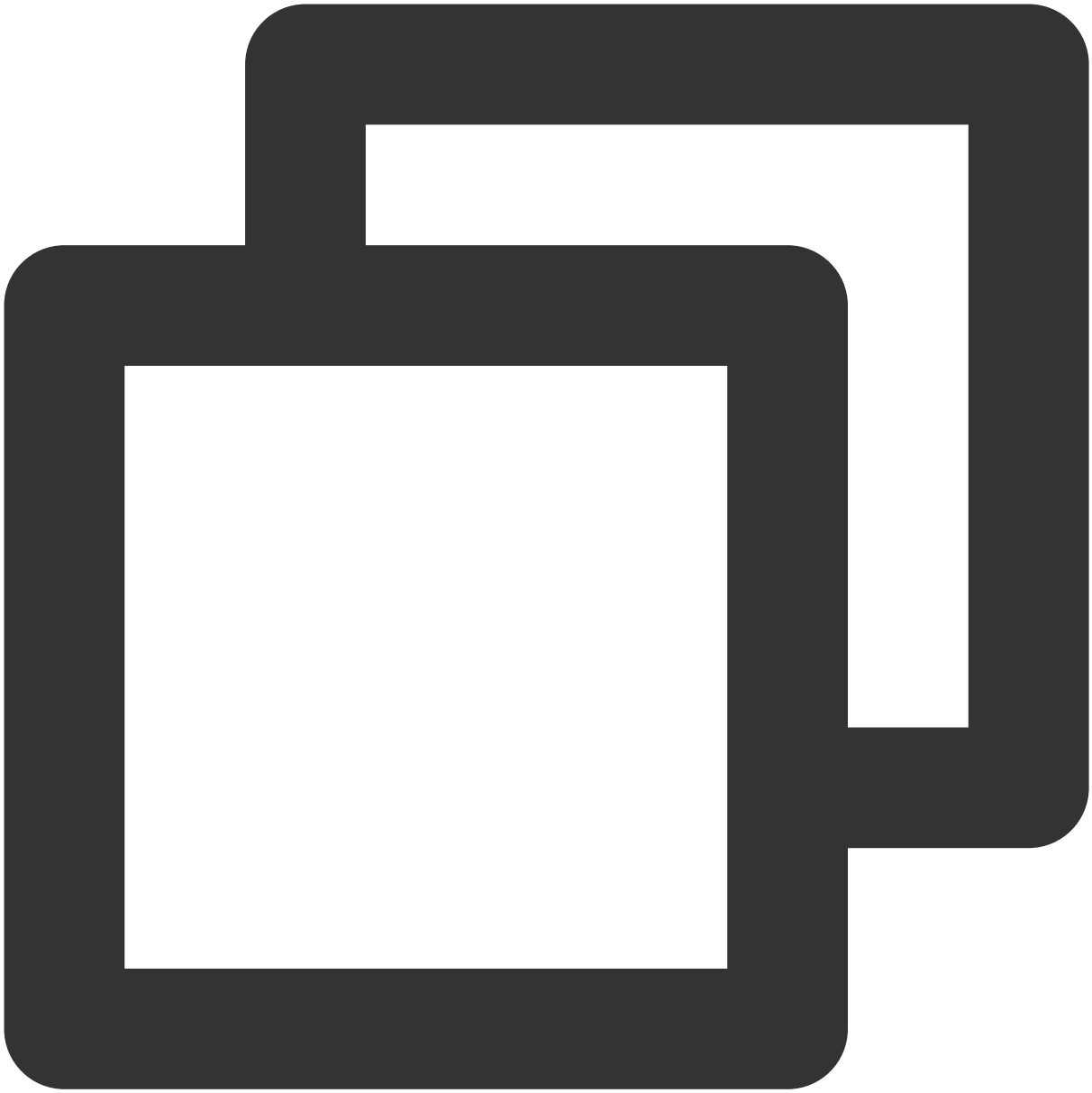
If your data is already in a self-built PostgreSQL database, you can use the `psql` command to easily migrate the data to PostgreSQL for Serverless.
The data migration is mainly divided into two steps:
1. Perform logical backup by using the `pg_dump` command to create dump data.
2. Restore the data backed up in the previous step to PostgreSQL for Serverless.

Prerequisites

You have prepared a PostgreSQL for Serverless instance. If not, please see Getting Started.

Step 1. Export data

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



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

Note:

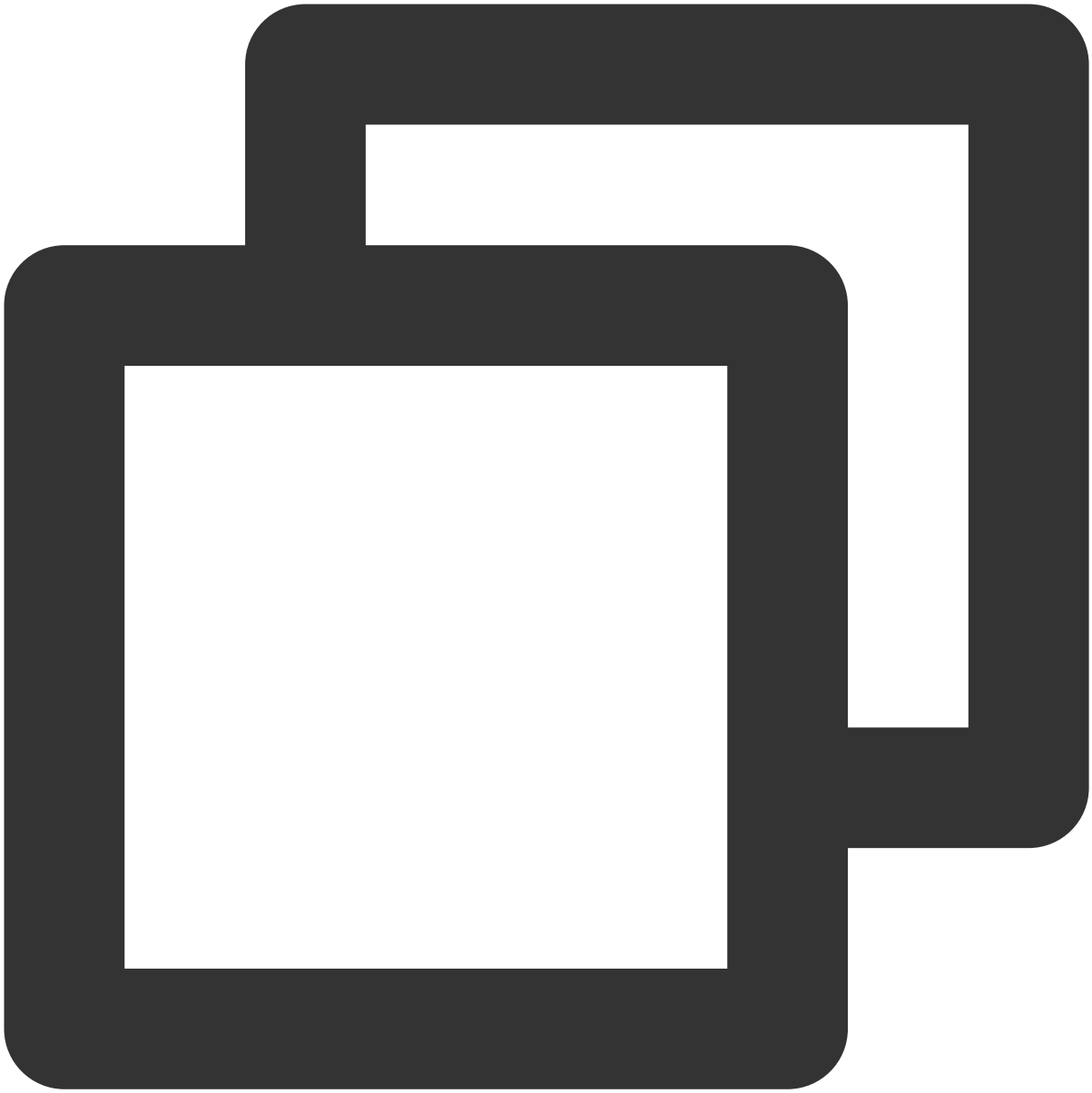
In order to avoid execution permission problems, you need to add the `-O` parameter.

Parameter	Description
username	Local database username
hostname	Local database host name

port	Local database port number
databasename	Name of the local database to be backed up
filename	Name of the backup file to be generated, such as mydump.sql

Step 2. Import data to PostgreSQL for Serverless

Preparations: upload the data backed up to a CVM instance in the same VPC as the PostgreSQL for Serverless instance, and then restore the data over the private network to ensure network stability and data security.
Log in to the CVM instance and run the following command in the PostgreSQL client to restore the data.



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

Note:

During the import, there may be some errors reported. You can find the specific causes according to the error messages. Some errors don't affect the data import.

Parameter	Description
username	PostgreSQL for Serverless database username
hostname	PostgreSQL for Serverless database address

desintationdb	PostgreSQL for Serverless database name
port	PostgreSQL for Serverless database port number
dumpfilename	Backup file name, such as mydump.sql