Copyright Notice

©2013-2019 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice

All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.
Contents

Best Practice
   Usage Specifications of TencentDB for MySQL
   Creating VPCs for TencentDB for MySQL
   Increase Business Load Capacity by MySQL Database
   Deploying Python Web Applications
   Build LAMP Stack for Web Application
   Build Drupal Website
   Configuring Application Automatic Reconnection
   Limits on Automatic Conversion from MyISAM to InnoDB
   Impact of Modifying MySQL Master Instance Parameters
   Call MySQL API by Python
      Purchase Instance
      Manage Instance
      Backup Task
Best Practice
Usage Specifications of TencentDB for MySQL

Last updated: 2020-04-01 16:46:10

Purpose

- To standardize the management and maintenance of TencentDB for MySQL to avoid unavailability and other issues caused by improper operations.
- To provide guidance for database developers on how to write SQL statements to ensure optimal performance of TencentDB for MySQL.

Permission Management Specifications

- To ensure the stability and security, permission restrictions are imposed on `super`, `shutdown`, and `file` in TencentDB for MySQL. The following error may occur when you run the `set` statement:

  #1227-Access denied; you need (at least one of) the SUPER privilege(s) for this operation

Solution: if you need to modify relevant parameters using "set", go to Database Management > Parameter Settings on the instance management page. If the parameters to be modified are not listed there, you can submit a ticket for application, and we will evaluate and make the modifications for you so as to ensure instance stability.

- Grant permission on demand. It is sufficient to grant general applications only the DML permissions (SELECT, UPDATE, INSERT, DELETE).
- Grant permission to users of general application at the database level.
- Allow authorized users to access only from specific IPs or IP range. This can be achieved by configuring security groups in the console as instructed there. To set a security group for public network access, be sure to allow all the egress IPs involved.
- Use different accounts for management and development.

Operation Specifications

Precautions
For enhanced instance security, do not use weak passwords.
For login over the private network, make sure that the CVM instance of the client and the TencentDB for MySQL instance are in the same region and under the same account.
If the binlogs downloaded from the console need to be parsed locally, make sure that the client's MySQL version is the same as that of the TencentDB for MySQL instance; otherwise, garbled characters will be displayed during parsing. It is recommended to use mysqlbinlog 3.4 or higher.
When downloading cold backup files to a CVM instance over the private network in the console, enclose the URL with quotation marks; otherwise, a 404 error will occur.

Suggestions
- Avoid performing online DDL operations during peak hours. You can use tools such as pt-online-schemachange.
- Avoid performing batch operations during peak hours.
- Avoid running an instance for multiple businesses so as to minimize the risk of mutual interference between businesses due to high coupling.
- It is recommended to disable automatic transaction committing and develop a habit of using begin; for online operations, which can help minimize the risk of data loss caused by faulty operations. In case of a faulty operation, you can use the rollback feature of TencentDB for MySQL for data restoration (rollback to any point in time in the last 5 days is supported). For tables without cross-database and cross-table logic, you can use quick or instant rollback for even faster data restoration. The new table after rollback is named "original table name_bak".
- For promotional campaigns of your business, make an estimate of the resources required in advance and optimize the instances. In case of a great demand for resources, contact your Tencent Cloud sales rep in a timely manner.

Database and Table Design Specifications

Precautions
- MyISAM and Memory engines are no longer supported in TencentDB for MySQL 5.6 or higher. If the Memory engine is required, you are recommended to use TencentDB for Redis or Memcached. When your self-built database is migrated to TencentDB for MySQL, MyISAM will be automatically converted to InnoDB during the migration.
- For a table containing an auto-increment column, a separate index must exist on the column. If composite indexing is used, the auto-increment column must be put in the first place.
- \texttt{row\_format} must be non-fixed.
- Each table must have a primary key. Even if no column is suitable for use as the primary key, you still have to add a meaningless column as the primary key. According to MySQL 1NF, a primary key...
value is saved on the standard InnoDB secondary index's leaf nodes. It is recommended to use a short auto-increment column as the primary key so as to reduce the disk capacity occupied by indexes and improve the efficiency. If `binbin_format` is row, deleting data in batches without the primary key can cause serious master-slave delay.

- Define fields as NOT NULL and set default values. NULL fields will cause unavailability of indexes, thus bringing problems to SQL development. NULL calculation can only be implemented based on IS NULL and IS NOT NULL.

**Suggestions**

- Plan the resources used by databases reasonably based on business scenario analysis and estimation of data access (including database read/write QPS, TPS, and storage). You can also configure various Cloud Monitor metrics for TencentDB for MySQL in the console.
- When building databases, put the tables for the same type of businesses into one database and try not to mix and match. Do not perform cross-database correlation operations in programs as doing so will affect subsequent quick rollbacks.
- Always use the utf8mb4 character set to minimize the risk of garbled characters. Some complex Chinese characters and emoji stickers can be displayed normally only in utf8mb4. If the character set is changed, the new character set will take effect only on tables created after the change. Therefore, it is recommended to select utf8mb4 as early as in the initialization of a new TencentDB for MySQL instance.
- For decimal fields, the decimal type is recommended. The float and double types have insufficient precision, especially for businesses involving money where the decimal type must be used.
- Do not use text/blob to store a large amount of text, binary data, images, files, and other contents in a database; instead, save such data as local disk files and only store their index information in the database.
- Avoid using foreign keys. It is recommended to implement the foreign key logic at the application layer. Foreign key and cascade update are not suitable for high-concurrency scenarios, because they may reduce the insertion performance and lead to deadlock in case of high concurrence.
- Reduce the coupling of business logic and data storage; use databases mainly for storing data and implement business logic at the application layer as much as possible; minimize the use of stored procedures, triggers, functions, events, views, and other advanced features due to their poor portability and scalability. If such objects exist in an instance, it is not recommended to set definer by default so as to avoid migration failures caused by inconsistency between migration account and definer.
- If you won't have a substantial business volume in the near future, do not use partition tables, which are mainly used for archive management in the courier and ecommerce industries. Do not rely on partition tables for performance enhancement, unless over 80% of queries in your business involve partition fields.
For business scenarios with a high read load and low requirement for consistency (data delay within seconds is acceptable), it is recommended to purchase read-only instances to implement read/write separation at the database level.

Index Design Specifications

Precautions

- Do not create indexes on the columns that are updated frequently and have a lower differentiation. Record updates will change the B+ tree, so creating indexes for frequently updated fields may greatly reduce the database performance.

- When creating a composite index, the index with the highest differentiation should be placed on the far left; for example, in `select xxx where a = x and b = x;`, if a and b are used together to create a composite index and a has higher differentiation, then the composite index should be created as `idx_ab(a, b)` If None-Equal To and Equal To conditions are used at the same time, the column with the Equal To condition must be put first; for example, for `where a xxx and b = xxx`, b must be placed at the forefront even if a has a higher differentiation, because index a will not be used in the query.

Suggestions

- It is recommended to use no more than 5 indexes in a single table and no more than 5 fields in a single index. Too many indexes may affect the filtering, occupy much more capacity, and consume more resources for management.

- Create indexes on the columns that are used for SQL filtering most frequently with a high cardinality value. It is meaningless to create indexes on a column not involved in SQL filtering. The higher the uniqueness of a field, the higher the cardinality value, and the better the index filtering result. Generally, an index column with a cardinality below 10% is considered an inefficient index, such as the gender field.

- When creating an index on the `varchar` field, it is recommended to specify an index length but not to index the entire column. This is because the `varchar` column is often long and specifying the index length can provide sufficient differentiation. Indexing the entire column will increase the maintenance costs. You can use `count (distinct left (column name, index length))/count (*)` to check index differentiation.

- Avoid using redundant indexes. If both index (a,b) and index (a) exist, (a) is considered a redundant index. If the query filtering is based on column a, the index (a,b) is sufficient.

- Use covering indexes reasonably to reduce IO overhead. In InnoDB, leaf nodes of a secondary index only save the values of their own keys and the primary key. If an SQL statement does not query such an index column or primary key, the query on the index will locate the corresponding
primary key first and then locate the desired column based on the primary key. This is TABLE ACCESS BY INDEX ROWID, which will incur extra IO overhead. Covering indexes can be used to solve this problem; for example, in `select a, b from xxx where a = xxx`, if `a` is not the primary key, a composite index can be created on `a` and `b` columns to prevent the problem.

**SQL Statement Writing Specifications**

**Precautions**

- Do not use LIMIT for UPDATE and DELETE operations, because LIMIT is random and may cause data errors; instead, you must use WHERE for such operations for exact match.
- Do not use `INSERT INTO t_xxx VALUES(xxx)` and the column attributes to be inserted must be specified explicitly to prevent data errors caused by changes in the table structure.
- The following are common reasons for invalid indexes in SQL statements:
  - Implicit type conversion; for example, if the type of index `a` is varchar and the SQL statement is `where a = 1;`, then varchar is changed to int.
  - Math calculations and functions are performed on the index columns; for example, date column is formatted using a function.
  - Columns on which a join operation is performed have different character sets.
  - Multiple columns have different sorting orders; for example, the index is `(a,b)`, but the SQL statement is `order by a b desc`.
  - When fuzzy queries are performed, some indexes can be queried for characters in the format of `xxx%`; however, in other cases, indexes will not be used.
  - Queries in reverse direction (such as "not", ",=", and "not in") are used.

**Suggestions**

- Ensure query on demand and reject `select *` to avoid the following problems:
  - The covering index does not work and the problem of TABLE ACCESS BY INDEX ROWID occurs, which leads to extra IO overhead.
  - Additional memory load; a large amount of cold data is imported to `innodb_buffer_pool_size` which may reduce the query hit rate.
  - Additional overhead in network transfer.
- Avoid using large transactions. It is recommended to split a large transaction into multiple small ones to avoid master-slave delay.
- Commit transactions in the business code in a timely manner to avoid unnecessary lock waits.
- Minimize the use of join operations for multiple tables and do not perform join operations on large tables. When a join operation is performed on two tables, the smaller one must be used as the
driving table, the columns to be joined must have the same character set, and all of them must have been indexed.

- Use LIMIT for paging optimization. The operation "LIMIT 80000, 10" is to filter out 80,010 records and then return the last 10 ones. This may cause a high load on the database. It is recommended to locate the first record before paging, such as 

```
SELECT * FROM test WHERE id = (SELECT id FROM test order by id LIMIT 80000,1 ) LIMIT 10 ;
```

- Avoid using an SQL statement with multi-level nested subqueries. The query optimizer prior to MySQL 5.5 can convert "in" to "exists" and does not go through the indexes. In this case, a large external table may result in poor performance.

It is difficult to totally avoid the above problems. The solution is not to use the aforementioned conditions as the primary filtering conditions; instead, set them as the conditions secondary to the primary filtering conditions for indexes.

- If a large number of full table scans is found in the monitor, set the `log_queries_not_using_indexes` parameter in the console and download the slow logs for analysis later. Do not keep it enabled for too long so as to avoid a surge of slow logs.

- Perform the required SQL audit before a business goes live. In daily OPS work, download slow query logs regularly for targeted optimization.
Creating VPCs for TencentDB for MySQL

Last updated: 2020-06-03 14:06:31

Tencent Cloud provides Virtual Private Cloud (VPC) for hosting TencentDB instances. You can launch Tencent Cloud resources such as TencentDB instances in a VPC.

A common scheme is to share data between a TencentDB instance and a web server running in the same VPC. This document uses this scheme to create a VPC and add a TencentDB instance to it. This document describes how to add CVM and TencentDB for MySQL instances in the same VPC to interconnect cloud resources over the private network.

Step 1. Create a VPC

A VPC has at least one subnet, and Tencent Cloud service resources can only be added in a subnet.

1. Log in to VPC Console.
2. Select the region of the VPC at the top of the list and click +New.
3. Enter the VPC information and original subnet information and click OK. The CIDRs of the VPC and subnet cannot be modified after creation.

   - The VPC CIDR can be any of the following IP ranges. For VPCs to communicate with each other over the private network, their CIDRs should not overlap.
     - 10.0.0.0 - 10.255.255.255 (mask range between 16 and 28)
     - 172.16.0.0 - 172.31.255.255 (mask range between 16 and 28)
     - 192.168.0.0 - 192.168.255.255 (mask range between 16 and 28)
   - The subnet CIDR must be within or the same as the VPC CIDR. For example, if the IP range of a VPC is 192.168.0.0/16, then that of its subnets can be...
Step 2. Create a subnet

You can create one or more subnets at a time.

1. Log in to VPC Console.
2. Click Subnet on the left sidebar to enter the management page.
3. Select the region and VPC in which the subnet is to be created and click +New.
4. Enter the subnet’s name, CIDR, availability zone, and associated route table.

5. (Optional) Click **Add a line** to create multiple subnets at a time.
6. Click **OK**.

**Step 3. Create a route table and associate it with a subnet**

You can create a custom route table, edit its routing policy, and associate it with a specified subnet. The route table associated with a subnet is used to specify the outbound routes for the subnet.

1. Log in to the VPC Console and select **Route Tables** on the left sidebar.
2. Select the region and VPC at the top of the list and click **New**.
3. In the pop-up dialog box, enter the name, network, routing rules and click **Create**. Return to the route table list to view the newly created route table.
4. Click **Subnet** on the left sidebar, select the subnet to be associated with the route table, and click **Change route table** in the **Operation** column to associate it.

**Step 4. Add a CVM instance**

1. Log in to [VPC Console](#).
2. Click **Subnet** on the left sidebar to enter the management page.
3. Click the “Add a CVM” icon in the row of the subnet where the CVM instance is to be added.

4. Complete the CVM purchase as instructed by the page. For more information, please see the [Purchasing Channels](#) documentation.

**Step 5. Add a TencentDB instance**

*New database*
1. Log in to the TencentDB for MySQL Console and click **Create** in the instance list to enter the purchase page.

2. In the **Network** section on the purchase page, select the previously created VPC and the corresponding subnet, and add the new TencentDB instance to the VPC.

---

**Existing database**

1. In the instance list, click an instance name or **Manage** in the "Operation" column to enter the instance details page.

2. In the **Network** section on the details page, switch to the corresponding VPC.
Increase Business Load Capacity by MySQL Database

Last updated : 2019-11-20 16:06:13

Databases with excellent performance and scalability can help you quickly increase the load capacity of your existing systems. With the same size of database, TencentDB for MySQL, if appropriately used, can significantly improve database concurrence for higher QPS.

1. Select a proper database configuration

1.1 Select the database version

TencentDB for MySQL is currently available in v5.5, v5.6, and v5.7, all of which are fully compatible with native MySQL. You are recommended to choose v5.6 and higher, as they use more stable database kernels, deliver better system performance by optimizing the design of v5.5 and lower, and come with a lot of appealing new features.

This document takes MySQL v5.7 as an example to illustrate the features of the new versions. This version is widely recognized for its impressive performance, reliability, and ease of use. Some of its improvements and new features are as shown below:

- **Native JSON support**
  In MySQL v5.7, a new data type has been added to store data in the native JSON format in MySQL tables, which has the following advantages:
  - **Document verification**: Only data segments in line with JSON rules can be written to JSON-type columns, which means that there is automated JSON syntax verification.
  - **Efficient access**: When a JSON document is stored in a JSON-type column, the data will not be stored as plain text; instead, it will be stored in the optimized binary format, so that its object members and array elements can be accessed more quickly.
  - **Performance enhancement**: An index can be created on data in JSON-type columns so as to improve the query performance. Such indexes can be implemented through the "function index" created on virtual columns.
  - **Convenience**: The inline syntax attached to JSON-type columns can be naturally integrated into document queries in SQL statements such as "features". "feature" is a JSON field:

```
SELECT feature->">properties.STREET" AS property_street FROM features WHERE id = 121254;
```
With MySQL v5.7, you can seamlessly integrate the best relational samples with the best
document samples in one tool so as to use the most appropriate ones out of them in different
applications and use cases, which greatly expands your range of applications.

- **SYS Schema**
  MySQL SYS Schema is a database schema consisting of a set of objects such as views, stored
  procedures, storage methods, tables, and triggers. It gives easy, readable, DBA- and developer-
  friendly access to the wealth of monitoring data stored in various tables in Performance Schema
  and INFORMATION_SCHEMA.
  It is included in MySQL v5.7 by default and provides summary views to answer the following
  common questions:
  - What is taking up all the resources of the database service?
  - Which CVM instance accesses the database server most frequently?
  - How is the instance memory used?

- **InnoDB improvements**
  - Online operations in InnoDB (Online DDL): You can dynamically adjust the buffer pool size to
    make it adaptive to the change of your business needs without restarting MySQL. InnoDB now
    can automatically empty its UNDO logs and table space online, thus eliminating one of the most
    common reasons for large shared table space files (ibdata1). In addition, MySQL v5.7 supports
    renaming indexes and changing the varchar size, both of which could be done only by
    recreating indexes or tables in previous versions.
  - InnoDB native partitioning: In MySQL v5.7, InnoDB includes the native support for partitioning,
    which can reduce the load and lower the memory usage by up to 90%.
  - InnoDB cache prefetching: When MySQL restarts, InnoDB will automatically retain 25% of the
    hottest data in the buffer pool, eliminating your need to preload or prefetch the data cache and
    preventing potential performance loss caused by MySQL restart.

For more information on improvements and new features in MySQL v5.7, see MySQL's official
documentation.

### 1.2 Select an instance specification (database memory)

Currently, TencentDB for MySQL doesn't offer separate CPU options; instead, the CPU will be
allocated proportionally according to the memory specification. You can purchase database
specifications based on your business characteristics. We have conducted thorough benchmark tests
on each type of instance so as to provide performance information for your reference when you
select specifications.

However, it should be noted that the sysbench-enabled tests cannot represent all business
scenarios. You are recommended to perform stress testing on your instance before launching it
officially, so that you can better understand how TencentDB for MySQL performs in your business scenario. For more information, see MySQL Performance Description.

Memory is one of the core instance metrics, which features an access speed much higher than that of a disk. Generally, the more data cached in the memory, the faster the database response. If the memory is small, after the stored data exceeds a certain amount, the excessive data will be stored to the disk. After that, when a new request accesses the data again, the data will be read from the disk into the memory, consuming disk IO and leading to slower database response.

For businesses with high read concurrence or sensitive to read delay, it is recommended to choose a higher memory specification so as to ensure high database performance.

1.3 Select a disk (data storage capacity)

The disk capacity of a TencentDB for MySQL instance includes only the MySQL data directories but not the logs such as binlog, relaylog, undolog, errorlog, and slowlog. When the amount of written data exceeds the instance disk capacity, if the instance is not upgraded, instance lock may be triggered. Therefore, when you purchase a disk, you are recommended to take into account the possible data volume increase in the future and select a larger disk, which helps prevent your instance from being locked or frequently upgraded due to insufficient disk capacity.

1.4 Select a proper data replication mode

TencentDB for MySQL provides three replication modes: async, semi-sync, and strong sync. For more information, see Database Instance Replication Mode. If your business is sensitive to write latency or database performance, you are recommended to choose the async replication mode.

1.5 High availability of TencentDB

High availability of TencentDB for MySQL is guaranteed by the master/slave and master/master architecture. Master-slave data sync is achieved through binlogs. In addition, the database can be rolled back to any previous point in time, which relies on backups and logs. Therefore, you generally do not need to set up a backup and restoration system on your own or pay additional fees to keep your instance highly available.

1.6 Scalability of TencentDB

All the different database versions and memory/disk specifications of TencentDB for MySQL support online dynamic hot upgrade. The upgrade process will not interrupt your business, eliminating your concerns over any database bottlenecks caused by business growth.

1.7 Use CVM and TencentDB for MySQL together
After a purchase is made, you generally need to use CVM and TencentDB for MySQL together. For more information, see Accessing TencentDB for MySQL from CVM.

2. Use a read-only instance as read extension

In common internet-based businesses, the read/write ratio of databases generally ranges from 4:1 to 10:1, which means that the read load of databases is much higher than the write load. When a performance bottleneck occurs, a common solution is to increase the read load. TencentDB for MySQL read-only instances are ideal for such issues. For more information, see Read-only Instance. Read-only instances can also be used for read-only access in various businesses; for example, the master instance undertakes read/write access for online businesses, while the read-only instance provides read-only query for internal businesses or data analysis platforms.

3. Disaster recovery scheme of TencentDB for MySQL

TencentDB for MySQL provides disaster recovery instances, helping you quickly set up remote disaster recovery for databases.

With the help of disaster recovery instances, multiple data centers in different regions can act as redundancy of each other, so that when one data center cannot provide a service due to failures or force majeure events, the service can be quickly switched to another data center. Disaster recovery instances use private network Direct Connect lines of Tencent Cloud to implement data sync, which can minimize the impact of delayed sync on your business when a disaster occurs. As long as the remote service logic is ready, the disaster recovery switchover can be completed in seconds.

4. 2-region-3-DC scheme

With TencentDB for MySQL, it only takes several simple steps to configure the 2-region-3-DC scheme:

- Purchase a TencentDB for MySQL intra-city strong-consistency cluster and select multi-AZ deployment (currently in beta test) which provides the 1-region-2-DC capacity.
- Add remote disaster recovery nodes to the cluster in order to build the 2-region-3-DC architecture.

5. Use disaster recovery instances to provide users with local access
A disaster recovery instance also adopts the high-availability master/slave and master/master architecture. In addition, it can be accessed in a read-only manner, which helps enable local access to your businesses for end users in different regions.
Deploying Python Web Applications

Last updated: 2020-06-03 14:06:32

Django is an open-source web application framework written in Python. This document describes how to deploy the default Django website to a CVM instance that runs Python 2.7.

Software environments used include CentOS 7.2, Python 2.7, and Django 1.11.

**Step 1. Log in to the CVM instance**

For more information on CVM purchase and access, please see Customizing Linux CVM Configurations.

**Step 2. Install Python**

Python is installed in CentOS by default. You can view the Python version by running `python --version`.

**Step 3. Install Django**

1. Install pip.

   `yum install python-pip`

2. Update the pip package.

   `pip install --upgrade pip`

3. Install Django through pip.

   `pip install Django==1.11`

4. View the Django version to see whether the installation is successful.

   ```
   python # Enter the Python command line
   >>> import django
   >>> django.VERSION
   ```

**Step 4. Install the `MySQLdb` module**

Install the supporting modules of MySQL.

```
yum install python-devel
yum install mysql-devel
```
yum -y install mysql-devel libxml2 libxml2-dev libxslt* zlib gcc openssl
yum install gcc libffi-devel python-devel openssl-devel
pip install MySQL-python

**Step 5. Install the Apache service**

1. Install Apache in the CVM instance with `yum`.

   ```
   yum install httpd -y
   ```

2. Launch the Apache service.

   ```
   service httpd start
   ```

3. Test Apache.

   In this step, you should configure an inbound rule with the source being **all** and the port protocol being **TCP:80** in the security group of your CVM instance. For more information on security group configuration, please see [Security Groups](#).

   Enter `http://xxx.xxx.xxx.xxx/` in your local browser (where `xxx.xxx.xxx.xxx` is the public IP of your CVM instance). If the following page appears, Apache has launched successfully.
Step 6. Install Apache's mod_wsgi extension as a Django application container

1. Install httpd-devel.

   ```
   yum install -y httpd-devel
   ```

2. Install mod_wsgi.

   ```
   yum install -y mod_wsgi
   ```

Step 7. Create a project to test the Django environment

1. Create a test project under `/usr/local` by running `django-admin.py startproject projectname`, where `projectname` is the project name.

   ```
   cd /usr/local
   django-admin.py startproject projectname
   ```

2. Create a `django.wsgi` file in the project's root directory to support Apache.

   ```
   cd /usr/local/projectname
   vim django.wsgi
   ```
3. Enter the following content in `django.wsgi`:

```python
import os
import sys
from django.core.wsgi import get_wsgi_application
sys.path.insert(0, os.path.join(os.path.dirname(os.path.realpath(__file__)), '..', '..'))
os.environ['DJANGO_SETTINGS_MODULE'] = 'projectname.settings'
application = get_wsgi_application()
```

4. Add Apache support by adding the following content to `httpd.conf` in `/etc/httpd/conf/httpd.conf`:

```conf
LoadModule wsgi_module modules/mod_wsgi.so
WSGIScriptAlias /python "/usr/local/projectname/django.wsgi"
<Directory "/usr/local/projectname">
   AllowOverride None
   Options None
   Require all granted
</Directory>
<Directory "/usr/local/projectname">
   AllowOverride None
   Options None
   Require all granted
</Directory>
```

5. Create a view and the `view.py` file in the **project directory** `/usr/local/projectname/projectname` as the access entry with the following content:

```python
from django.http import HttpResponse
def hello(request):
    return HttpResponse("Hello world !")
```

6. Configure the URL and the `urls.py` file in the **project directory** `/usr/local/projectname/projectname`. Delete the original content and add the following:

```python
from django.conf.urls import *
from projectname.view import hello
urlpatterns = [
    url(r'^hello/$', hello),
]
```

7. Modify the `settings.py` file in the **project directory** `/usr/local/projectname/projectname`.

```python
ALLOWED_HOSTS = ['*']
```

8. Restart the Apache service.

```bash
service httpd restart
```
9. Enter `http://xxx.xxx.xxx.xxx/python/hello` in your local browser (where `xxx.xxx.xxx.xxx` is the public IP of your CVM instance). If “Hello world!” appears on the page, the project environment has been set up successfully.

**Step 8. Configure TencentDB in Django (optional)**

1. Configure the `settings.py` file in the project directory.

   ```python
   DATABASES = {
       'default': {
           'ENGINE': 'django.db.backends.mysql',
           'NAME': 'mysql',
           'USER': 'root',  # TencentDB account name
           'PASSWORD': '123456',  # TencentDB account password
           'HOST': '0.0.0.0',  # TencentDB private IP address
           'PORT': '3306',  # TencentDB port
       }
   }
   ```

2. After configuration, run the following command to test the database connection:

   ```bash
   $python manage.py validate/check
   ```

3. Once the test is passed, database operations can be performed. For more information, please see [Models and databases](#).
Build LAMP Stack for Web Application

Last updated: 2019-11-20 16:03:15

LAMP (Linux + Apache + MySQL/MariaDB + Perl/PHP/Python) is a set of open-source software programs often used to set up dynamic websites or servers. These independent programs are usually used together and increasingly compatible with one another to form a powerful web application platform.

This tutorial guides you through the following process: starting a TencentDB instance and configuring a LAMP application with a CVM instance to connect to the high-availability environment of the TencentDB instance.

The database can be separated from the environment lifecycle after you run the TencentDB instance. This allows you to connect multiple servers to the same database for simplified database OPS, eliminating your concerns over database installation, deployment, version update, and troubleshooting.

---

The TencentDB and CVM instances used in the tutorial reside in the same region. If this is not the case, see Access via Public Network.

---

**Initializing the TencentDB Instance**

For more information on how to purchase and initialize TencentDB instances, see Purchase Method and Initializing a TencentDB for MySQL Instance.

**Logging in to the CVM Instance**

For more information on how to purchase and access CVM instances, see Getting Started with Linux-based CVM. CentOS is used in this tutorial.

**Installing the MySQL Client**

1. Install the MySQL client to the CVM instance with `yum`.

   ```bash
   yum install mysql -y
   ```
2. Connect to the TencentDB instance after the installation is completed.

```
mysql -h hostname -u username -p
```

Here, "hostname" is the private IP of the TencentDB instance and "username" is the username of your database.

3. After the connection is successful, you can close the instance and proceed to the next step.
Installing the Apache Service

1. Install Apache in the CVM instance with `yum`.

```bash
yum install httpd -y
```

![Install Apache Command Output](image)

2. Start the Apache service.

```bash
service httpd start
```

3. Test Apache.
In this step, you should configure an inbound rule with the source being **all** and the port protocol being **TCP:80** in the security group of your CVM instance. For more information on how to configure the security group, see [Security Group](#).

Enter `http://115.xxx.xxx.xxx/` in your local browser (where `115.xxx.xxx.xxx` is the public IP of your CVM instance). If the following page appears, Apache has started successfully.

![Testing 123](#)

**Installing PHP**

1. Install PHP in the CVM instance with `yum`.

```
yum install php -y
```
Creating a Project to Test the LAMP Environment

1. Create an info.php file in the `/var/www/html` directory of the CVM instance. Below is the sample code:

   ```php
   <?php phpinfo(); ?>
   ```

2. Restart the Apache service.

   ```
   service httpd restart
   ```
3. Enter `http://0.0.0.0/info.php` in your local browser (where **0.0.0.0** is the public IP of your CVM instance). If the following page appears, the LAMP service has been deployed successfully.

```
PHP Version 5.4.16

<table>
<thead>
<tr>
<th>System</th>
<th>Linux VM_165_193_centos 3.10.0-327.36.3.el7.x86_64 #1 SMP Mon Oct 24 15:09:20 UTC 2016 x86_64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Date</td>
<td>Nov 6 2016 00:30:05</td>
</tr>
<tr>
<td>Server API</td>
<td>Apache 2.0 Handler</td>
</tr>
<tr>
<td>Virtual Directory Support</td>
<td>disabled</td>
</tr>
<tr>
<td>Configuration File (php.ini) Path</td>
<td>/etc</td>
</tr>
<tr>
<td>Loaded Configuration File</td>
<td>/etc/php.ini</td>
</tr>
<tr>
<td>Scan this dir for additional .ini files</td>
<td>/etc/php.d</td>
</tr>
<tr>
<td>PHP API</td>
<td>20100412</td>
</tr>
<tr>
<td>PHP Extension</td>
<td>20100525</td>
</tr>
<tr>
<td>Zend Extension</td>
<td>220100525</td>
</tr>
<tr>
<td>Zend</td>
<td>AP2020100525-NTS</td>
</tr>
</tbody>
</table>
```
Build Drupal Website

Last updated: 2020-02-25 18:06:24

Drupal is an open-source content management framework written in PHP, which consists of a content management system and a PHP development framework. It can be used to build rich-featured dynamic websites ranging from personal blogs to large communities. This tutorial describes how to build a Drupal ecommerce website on a CVM instance. Software environments used here include CentOS v7.2, Drupal v7.56, and PHP v5.4.16.

Logging in to the CVM Instance

For more information on how to purchase and access CVM instances, see Getting Started with Linux-based CVM.

Installing the MariaDB Service

1. MariaDB is supported by CentOS v7 and higher by default, so it is used here. Install the MariaDB service in the CVM instance with `yum`.

   ```
   yum install mariadb-server mariadb -y
   ```

2. Start the MariaDB service.

   ```
   systemctl start mariadb
   ```

3. Create a database named drupal for Drupal.

   ```
   mysqladmin -u root -p create drupal
   ```

   Here, "drupal" is the database name used in the Drupal service.

4. Create a user for the database.

   ```
   mysql -u root -p
   ```

   Authorize the user and exit the database after the authorization is successful.

   ```
   GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER, CREATE TEMPORARY TABLES, LOCK TABLES ON drupal.* TO 'username'@'localhost' IDENTIFIED BY 'password';
   FLUSH PRIVILEGES;
   exit
   ```

   Here, "username" and "password" are the database username and password used in the Drupal service, respectively.
Installing the Apache Service

1. Install Apache in the CVM instance with `yum`.
   
   ```bash
   yum install httpd -y
   ```

2. Start the Apache service.
   
   ```bash
   service httpd start
   ```

3. Test Apache.
   
   In this step, you should configure an inbound rule with the source being `all` and the port protocol being `TCP:80` in the security group of your CVM instance. For more information on how to configure the security group, see Security Group.

   Enter `http://115.xxx.xxx.xxx/` in your local browser (where `115.xxx.xxx.xxx` is the public IP of your CVM instance). If the following page appears, Apache has started successfully.

   ![Testing 123](image)

   **This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page it means that this site is working properly. This server is powered by CentOS.**

   **Just visiting?**
   The website you just visited is either experiencing problems or is undergoing routine maintenance.
   If you would like to let the administrators of this

   **Are you the Administrator?**
   You should add your website content to the directory `/var/www/html/.
   To prevent this page from ever being used, follow the instructions in the file
   `/etc/httpd/conf.d/welcome.conf`

Installing PHP

1. Install PHP and its extensions in the CVM instance with `yum`.
yum install php php-dom php-dg php-mysql php-pdo -y

2. Create an info.php file in the /var/www/html directory of the CVM instance to check whether PHP is successfully installed. Below is the sample code:

```php
<?php phpinfo(); ?>
```

3. Restart the Apache service.

```
service httpd restart
```

4. Enter `http://115.xxx.xxx.xxx/info.php` in your local browser (where 115.xxx.xxx.xxx is the public IP of your CVM instance). If the following page appears, PHP has been installed successfully.

![PHP Version 5.4.16](image)

#### Installing the Drupal Service

1. Download the Drupal installation package.
2. Decompress the package to the root directory of the website.

```bash
unzip drupal-7.56.zip
mv drupal-7.56/* /var/www/html/
```

3. Download the translation kit.

```bash
cd /var/www/html/
```

4. Modify the owner and group to which the `sites` directory belongs.

```bash
chown -R apache:apache /var/www/html/sites
```

5. Restart the Apache service.

```bash
service httpd restart
```

6. Enter `http://115.xxx.xxx.xxx/` in your local browser (where `115.xxx.xxx.xxx` is the public IP of your CVM instance) to go to the installation interface of Drupal, select the version to be installed,
and click **Save and continue**.

7. Select the language for installation and click **Save and continue**.

8. Set up the database and enter the database information you configured when *installing the MariaDB service*.

9. Enter the site information.

0. Complete the process of Drupal installation.

Configuring Application Automatic Reconnection

Last updated : 2020-06-15 12:58:19

This document describes the impact of disconnection during instance switch and how to configure automatic reconnection.

Background

When you are adjusting database instance specification, upgrading database engine, the master instance is hanging due to overload, the hardware fails, etc., TencentDB for MySQL instance may need to switch, causing disconnection for few seconds.

If automatic reconnection is not configured, the application will disconnect after the master/slave switch and normal business access will be affected.

We recommend that you configure automatic reconnection for applications and switch instances during the maintenance window.

Configuring Automatic Reconnection

To avoid application connection exceptions due to master/slave switch, we recommend that you configure automatic reconnection for TencentDB for MySQL applications by configuring the connection pool parameters, i.e., `connectTimeOut` and `socketTimeOut`.

Configure parameter values according to business scenarios. For OLTP (On-Line Transaction Processing) business scenarios, both parameters should be configured as 20 seconds.

- `connectTimeOut`: timeout period for the application to establish a TCP connection with the database server. We recommend that you configure this parameter with a value greater than the response time between the application and the database server.
- `socketTimeOut`: timeout period while waiting for a response after packets are sent over the TCP connection. We recommend that you configure this parameter to the maximum execution time for a single SQL statement.
Limits on Automatic Conversion from MyISAM to InnoDB

This document describes how to troubleshoot the table creation error when the MyISAM storage engine is automatically converted to InnoDB.

Background

TencentDB for MySQL supports InnoDB storage engine by default. In MySQL 5.6 and later versions, MyISAM and Memory engines are no longer supported. For more information, please see Database Storage Engines.

When the database is migrated or upgraded to TencentDB for MySQL 5.6 or above, the system will automatically convert MyISAM to InnoDB.

Contrary to MyISAM, InnoDB does not support a composite primary key that includes an AUTO_INCREMENT column. Therefore, when you create a table after MyISAM has been converted to InnoDB, an error will be reported as follows: ERROR 1075 (42000):Incorrect table definition; there can be only one auto column and it must be defined as a key.

To create a composite primary key that includes an AUTO_INCREMENT column in InnoDB, we recommend that you add an index for the AUTO_INCREMENT column.

Solution

1. The SQL statement that triggers a table creation error is as follows:

```sql
create table t_complexkey
(
  id int(8) AUTO_INCREMENT,
  name varchar(19),
  value varchar(10),
  primary key (name, id)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```
The error is reported as follows:

```
ERROR 1075 (42000): Incorrect table definition, there can be only one auto column and it must be defined as a key
```

2. Add an index and modify the SQL statement as follows:

```sql
create table t_complexkey
(
id int(8) AUTO_INCREMENT,
name varchar(19),
value varchar(10),
primary key (name, id),
key key_id (id)  ## Add an index for the AUTO_INCREMENT column
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

The table is created successfully as follows:

```
Query OK, 0 rows affected, 1 warning (0.01 sec)
```

3. Run the following command to query the table structure:

```sql
show create table t_complexkey;
```
```sql
MySQL [huawu]>
show create table t_complexkey;

+--------------------------+
| Table | Create Table |
+--------------------------+
| t_complexkey | CREATE TABLE `t_complexkey` ( |
|     |   `id` int(8) NOT NULL AUTO_INCREMENT, |
|     |   `name` varchar(19) NOT NULL, |
|     |   `value` varchar(10) DEFAULT NULL, |
|     |   PRIMARY KEY (`name`, `id`), |
|     |   KEY `key_id` (`id`) |
|     |   ENGINE=InnoDB DEFAULT CHARSET=utf8 |
|     | ) |
| row in set (0.00 sec) |
Impact of Modifying MySQL Master Instance Parameters

Last updated: 2019-11-20 16:10:45

For TencentDB for MySQL, you can modify the parameters of a master instance in the console. Modifying some crucial parameters in an improper way will lead to exceptions in disaster recovery instances and data inconsistency. This document describes the consequences of modifying the following crucial parameters.

**lower_case_table_names**

**Default value:** 0

**Role:** When creating a database or table, you can set whether storage and query operations are case-sensitive. This parameter can be set to 0 (case-sensitive) or 1 (case-insensitive), and the default value is 0.

**Impact:** After the parameters of the master instance are modified, the parameters of the disaster recovery instance cannot be modified accordingly, as the master instance is set as case-sensitive, but the disaster recovery instance is not; for example, if two tables named "Test" and "TEst" are created in the master instance, then data sync will fail when the disaster recovery instance uses the corresponding logs, because the table name "TEst" already exists.

**auto_increment_increment**

**Default value:** 1

**Role:** It is used as the increment value of the auto-increment column AUTO_INCREMENT. Its value can range from 1 (default value) to 65,535.

**Impact:** After the parameters of the master instance are modified, the parameters of the disaster recovery instance cannot be modified accordingly. When the increment value is modified for the master instance but not for the disaster recovery instance, master-slave data inconsistency will occur.

**auto_increment_offset**

**Default value:** 1

**Role:** It is used as the start value (offset) of the auto-increment column AUTO_INCREMENT. Its value can range from 1 (default value) to 65,535.

**Impact:** After the parameters of the master instance are modified, the parameters of the disaster recovery instance cannot be modified accordingly. When the offset value is modified for the master instance but not for the disaster recovery instance, master-slave data inconsistency will occur.
sql_mode

Default value: NO_ENGINE_SUBSTITUTION

Role: TencentDB for MySQL can operate in different SQL modes, which define the SQL syntax and data check that it should support. The default value of this parameter in v5.6 is NO_ENGINE_SUBSTITUTION, which means that if the used storage engine is disabled or not compiled, an error will be thrown; in v5.7, the default value is

ONLY_FULL_GROUP_BY, STRICT_TRANS_TABLES, NO_ZERO_IN_DATE, NO_ZERO_DATE, ERROR_FOR_DIVISION_BY_ZERO, NO_AUTO_CREATE_USER, NO_ENGINE_SUBSTITUTION.

Here:

- **ONLY_FULL_GROUP_BY** means that in a GROUP BY operation, the column in SELECT or the HAVING or ORDER BY subquery must be a function column that appears in or relies on GROUP BY;
- **STRICT_TRANS_TABLES** enables strict mode;
- **NO_ZERO_IN_DATE** indicates whether the month and day of a date can contain 0 and is subject to the status of the strict mode;
- **NO_ZERO_DATE** means that dates in the database cannot contain zero date and is subject to the status of the strict mode;
- **ERROR_FOR_DIVISION_BY_ZERO** means that in strict mode, if data is divided by 0 during the INSERT or UPDATE process, an error rather than a warning will be thrown, while in non-strict mode, NULL will be returned;
- **NO_AUTO_CREATE_USER** prohibits GRANT from creating a user whose password is empty;
- **NO_ENGINE_SUBSTITUTION** means that if the used storage engine is disabled or not compiled, an error will be thrown.

Impact: After the parameters of the master instance are modified, the parameters of the disaster recovery instance cannot be modified accordingly. When the SQL mode is changed for the master instance but not for the disaster recovery instance, for example, if the SQL mode limit of the master instance is smaller than that of the disaster recovery instance, the SQL statements that are successfully executed in the master instance may trigger errors when synced to the disaster recovery instance and thus lead to master-slave data inconsistency.
Call MySQL API by Python
Purchase Instance

Last updated : 2019-11-20 15:57:15

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateDBInstanceHour</td>
<td>Creates a pay-as-you-go TencentDB instance</td>
</tr>
<tr>
<td>DescribeDBInstances</td>
<td>Queries the list of instances</td>
</tr>
<tr>
<td>DescribeDBPrice</td>
<td>Inquires the price of a TencentDB instance</td>
</tr>
<tr>
<td>DescribeDBZoneConfig</td>
<td>Queries the specifications of purchasable TencentDB instances</td>
</tr>
<tr>
<td>InitDBInstances</td>
<td>Initializes new instances</td>
</tr>
</tbody>
</table>

CreateDBInstanceHour for Creating a Pay-as-you-go TencentDB Instance

'''Hourly billing requires freezing an amount in your account, so If your account balance is 0, no purchase can be made'''

#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-beijing")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.CreateDBInstanceHourRequest()
    req.EngineVersion = "5.6"
    req.Zone = "ap-beijing-3"
    req.ProjectId = 0

Call MySQL API by Python
Purchase Instance

Last updated : 2019-11-20 15:57:15

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateDBInstanceHour</td>
<td>Creates a pay-as-you-go TencentDB instance</td>
</tr>
<tr>
<td>DescribeDBInstances</td>
<td>Queries the list of instances</td>
</tr>
<tr>
<td>DescribeDBPrice</td>
<td>Inquires the price of a TencentDB instance</td>
</tr>
<tr>
<td>DescribeDBZoneConfig</td>
<td>Queries the specifications of purchasable TencentDB instances</td>
</tr>
<tr>
<td>InitDBInstances</td>
<td>Initializes new instances</td>
</tr>
</tbody>
</table>
req.GoodsNum = 1
req.Memory = 1000
req.Volume = 50
req.InstanceRole = "master"
req.Port = 3311
req.Password = "CDB@Qcloud"
req.ParamList = [{"name":"max_connections","value":"1000"},{"name":"lower_case_table_names","value":"1"}]
req.ProtectMode = 1
req.SlaveZone = "ap-beijing-3"
req.InstanceName = "oneday1"
req.AutoRenewFlag = 0

# Call the API you want to access through the client object. You need to pass in the request object
resp = client.CreateDBInstanceHour(req)

# A string return packet in JSON format is outputted
print(resp.to_json_string())
except TencentCloudSDKException as err:
    msg = traceback.format_exc()  # Method 1
    print(msg)

DescribeDBInstances for Querying the List of Instances

#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
req = models.DescribeDBInstancesRequest()
req.EngineVersions = ["5.6"]
req.OrderBy = "instanceId"
req.InstanceIds = ["cdb-1j8lumf6"]

# Call the API you want to access through the client object. You need to pass in the request object
resp = client.DescribeDBInstances(req)

# A string return packet in JSON format is outputted
print(resp.to_json_string())
except TencentCloudSDKException as err:
    msg = traceback.format_exc()  # Method 1
    print(msg)

---

DescribeDBPrice for Inquiring the Price of a TencentDB Instance

#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-guangzhou")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBPriceRequest()
    req.Zone = "ap-guangzhou-3"
    req.GoodsNum = 1
    req.Memory = 2000
    req.Volume = 1000
    req.PayType = 'PRE_PAID'
    req.Period = 1

    # Call the API you want to access through the client object. You need to pass in the request object
    resp = client.DescribeDBPrice(req)

    # A string return packet in JSON format is outputted
DescribeDBZoneConfig for Querying the Specifications of Purchasable TencentDB Instances

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBZoneConfigRequest()
    #req.InstanceId = "cdb-j0edpju5"

    # Call the API you want to access through the client object. You need to pass in the request object
    resp = client.DescribeDBZoneConfig(req)

    # A string return packet in JSON format is outputted
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

InitDBInstances for Initializing New Instances

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
```
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.InitDBInstancesRequest()
    req.InstanceIds = ["cdb-c752yqcn"]
    req.NewPassword = "CDB@Qcloud"

    req.Parameters = [{'name':"max_connections","value":"100"},{'name':"character_set_server","value":"utf8"},{"name":"lower_case_table_names","value":"1"}]

    # Call the API you want to access through the client object. You need to pass in the request object
    resp = client.InitDBInstances(req)

    # A string return packet in JSON format is outputted
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
Manage Instance

Last updated: 2019-11-20 15:58:19

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModifyInstanceParam</td>
<td>Modifies instance parameters</td>
</tr>
<tr>
<td>CloseWanService</td>
<td>Disables public network access for an instance</td>
</tr>
<tr>
<td>OpenWanService</td>
<td>Enables public network access for an instance</td>
</tr>
<tr>
<td>RestartDBInstances</td>
<td>Restarts instances</td>
</tr>
<tr>
<td>OpenDBInstanceGTID</td>
<td>Enables GTID for an instance</td>
</tr>
<tr>
<td>ModifyDBInstanceName</td>
<td>Renames a TencentDB instance</td>
</tr>
<tr>
<td>ModifyDBInstanceProject</td>
<td>Modifies the project to which a TencentDB instance belongs</td>
</tr>
<tr>
<td>ModifyDBInstanceVipVport</td>
<td>Modifies the IP and port number of a TencentDB instance</td>
</tr>
<tr>
<td>DescribeDBInstanceCharset</td>
<td>Queries the character set of a TencentDB instance</td>
</tr>
<tr>
<td>DescribeDBInstanceConfig</td>
<td>Queries the configuration information of a TencentDB instance</td>
</tr>
<tr>
<td>DescribeDBInstanceGTID</td>
<td>Queries whether GTID is activated for a TencentDB instance</td>
</tr>
<tr>
<td>DescribeDBInstanceRebootTime</td>
<td>Queries the estimated restart time of a TencentDB instance</td>
</tr>
</tbody>
</table>

ModifyInstanceParam for Modifying Instance Parameters

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
```
cred = credential.Credential("secretId", "secretKey")

# Instantiate the client object to request the product (with TencentDB as an example)
client = cdb_client.CdbClient(cred, "ap-shanghai")

# Instantiate a request object
req = models.ModifyInstanceParamRequest()
req.InstanceIds = ["cdb-ly6g3zj8", "cdb-7ghaiocc"]
req.ParamList = [{"name":"max_connections","currentValue":"100"},{"name":"character_set_server", "currentValue":"utf8"},{"name":"lower_case_table_names", "currentValue":"1"}]
#req.ParamList = [{"name":"max_connections","currentValue":"100"}]
#param = models.Parameter()
#param.Name = "max_connections"
#paramCurrentValue = "1000"
#req.ParamList = [param]

print req
# Call the API you want to access through the client object. You need to pass in the request object
resp = client.ModifyInstanceParam(req)

# A string return packet in JSON format is outputted
print(resp.to_json_string())
except TencentCloudSDKException as err:
    msg = traceback.format_exc()
    # Method 1
    print (msg)

---

CloseWanService for Disabling Public Network Access for an Instance

#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

# Instantiate a request object: 
```python
req = models.ModifyInstanceParamRequest()
```

```python
req = models.CloseWanServiceRequest()
req.InstanceId = "cdb-1y6g3zj8"
```

# Call the API you want to access through the client object. You need to pass in the request object
```python
resp = client.CloseWanService(req)
```

# A string return packet in JSON format is outputted
```python
print(resp.to_json_string())
```

```python
except TencentCloudSDKException as err:
    print(err)
```

### OpenWanService for Enabling Public Network Access for an Instance

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to
    # be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: 
```
```python
req = models.OpenWanServiceRequest()
```
```python
req.InstanceId = "cdb-1y6g3zj8"
```

# Call the API you want to access through the client object. You need to pass in the request object
```python
resp = client.OpenWanService(req)
```

# A string return packet in JSON format is outputted
```python
print(resp.to_json_string())
```
```python
except TencentCloudSDKException as err:
    print(err)
```

### RestartDBInstances forRestarting Instances
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module

from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, ”ap-shanghai”)

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.RestartDBInstancesRequest()
    req.InstanceIds = [”cdb-7ghaiocce”]

    # Call the API you want to access through the client object. You need to pass in the request object
    resp = client.RestartDBInstances(req)

    # A string return packet in JSON format is outputted
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)

---

OpenDBInstanceGTID for Enabling GTID for an Instance

#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module

from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")
ModifyDBInstanceName for Renaming a TencentDB Instance

#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")
    
    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-beijing")
    
    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.ModifyDBInstanceNameRequest()
    req.InstanceId = "cdb-cukm86n2"
    req.InstanceName = "1s Chinese"
    
    # Call the API you want to access through the client object. You need to pass in the request object
    resp = client.ModifyDBInstanceName(req)
    
    # A string return packet in JSON format is outputted
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
except TencentCloudSDKException as err:
    print(err)

ModifyDBInstanceProject for Modifying the Project to Which a TencentDB Instance Belongs

#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

def DescribeDBInstancesList():
    try:
        # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
        cred = credential.Credential("secretId", "secretKey")

        # Instantiate the client object to request the product (with TencentDB as an example)
        client = cdb_client.CdbClient(cred, "ap-shanghai")

        # Instantiate a request object: req = models.ModifyInstanceParamRequest()
        req = models.ModifyDBInstanceProjectRequest()
        req.InstanceIds = ["cdb-7ghaiocc"]
        req.NewProjectId = 1

        # Call the API you want to access through the client object. You need to pass in the request object
        resp = client.ModifyDBInstanceProject(req)

        # A string return packet in JSON format is outputted
        print(resp.to_json_string())
    except TencentCloudSDKException as err:
        print(err)

msg = traceback.format_exc()
# Method 1
print(msg)
ModifyDBInstanceVipVport for Modifying the IP and Port Number of a TencentDB Instance

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to
    # be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.ModifyDBInstanceVipVportRequest()
    req.InstanceId = "cdb-7ghaiocc"
    req.DstIp = "10.0.0.13"
    req.DstPort = 1025
    req.UniqVpcId = 1111

    # Call the API you want to access through the client object. You need to pass in the request object
    resp = client.ModifyDBInstanceVipVport(req)

    # A string return packet in JSON format is outputted
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    msg = traceback.format_exc()
    print(msg)
```

DescribeDBInstanceCharset for Querying the Character Set of a TencentDB Instance
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module

from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBInstanceCharsetRequest()
    req.InstanceId = "cdb-1y6g3zj8"

    # Call the API you want to access through the client object. You need to pass in the request object
    resp = client.DescribeDBInstanceCharset(req)

    # A string return packet in JSON format is outputted
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)

DescribeDBInstanceConfig for Querying the Configuration Information of a TencentDB Instance

#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module

from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBInstanceCharsetRequest()
    req.InstanceId = "cdb-1y6g3zj8"
cred = credential.Credential("secretId", "secretKey")

# Instantiate the client object to request the product (with TencentDB as an example)
client = cdb_client.CdbClient(cred, "ap-shanghai")

# Instantiate a request object: req = models.ModifyInstanceParamRequest()
req = models.DescribeDBInstanceConfigRequest()
req.InstanceId = "cdb-1y6g3zj8"

# Call the API you want to access through the client object. You need to pass in the request object
resp = client.DescribeDBInstanceConfig(req)

# A string return packet in JSON format is outputted
print(resp.to_json_string())

except TencentCloudSDKException as err:
    print(err)


DescribeDBInstanceGTID for Querying Whether GTID Is Activated for a TencentDB Instance

#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBInstanceGTIDRequest()
    req.InstanceId = "cdb-1y6g3zj8"

    # Call the API you want to access through the client object. You need to pass in the request object
    resp = client.DescribeDBInstanceGTID(req)

    # A string return packet in JSON format is outputted
    print(resp.to_json_string())

except TencentCloudSDKException as err:
    print(err)
resp = client.DescribeDBInstanceGTID(req)

# A string return packet in JSON format is outputted
print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)

DescribeDBInstanceRebootTime for Querying the Estimated Restart Time of a TencentDB Instance

#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to
    # be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBInstanceRebootTimeRequest()
    req.InstanceIds = ["cdb-ly6g3zj8"]

    # Call the API you want to access through the client object. You need to pass in the request object
    resp = client.DescribeDBInstanceRebootTime(req)

    # A string return packet in JSON format is outputted
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
Backup Task
Last updated: 2019-11-20 15:59:25

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateBackup</td>
<td>Creates a TencentDB instance backup</td>
</tr>
<tr>
<td>DeleteBackup</td>
<td>Deletes a TencentDB instance backup</td>
</tr>
<tr>
<td>DescribeBackupConfig</td>
<td>Queries the configuration information of a TencentDB instance backup</td>
</tr>
<tr>
<td>DescribeBackupDatabases</td>
<td>Queries the list of backed up databases</td>
</tr>
<tr>
<td>DescribeBackupTables</td>
<td>Queries backup data tables of the specified database</td>
</tr>
<tr>
<td>DescribeBackups</td>
<td>Queries backup logs</td>
</tr>
<tr>
<td>DescribeBinlogs</td>
<td>Queries binary logs</td>
</tr>
<tr>
<td>DescribeSlowLogs</td>
<td>Queries slow logs</td>
</tr>
<tr>
<td>ModifyBackupConfig</td>
<td>Modifies the database backup configuration</td>
</tr>
</tbody>
</table>

CreateBackup for Creating a TencentDB Instance Backup

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.CreateBackupRequest()
```
req.InstanceId = "cdb-7ghaiocc"
req.BackupMethod = "logical"

print req
# Call the API you want to access through the client object. You need to pass in the request object
resp = client.CreateBackup(req)

# A string return packet in JSON format is outputted
print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)

---

DeleteBackup for Deleting a TencentDB Instance Backup

#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module

from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.DeleteBackupRequest()

    req.InstanceId = "cdb-7ghaiocc"
    #print req.BackupId
    req.BackupId = 105119782

    # Call the API you want to access through the client object. You need to pass in the request object
    resp = client.DeleteBackup(req)

    # A string return packet in JSON format is outputted
print(resp.to_json_string())
except TencentCloudSDKException as err:
print(err)

## DescribeBackupConfig for Querying the Configuration Information of a TencentDB Instance Backup

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.DescribeBackupConfigRequest()
    req.InstanceId = "cdb-7ghaiocc"

    print req
    # Call the API you want to access through the client object. You need to pass in the request object
    resp = client.DescribeBackupConfig(req)

    # A string return packet in JSON format is outputted
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

## DescribeBackupDatabases for Querying the List of Backed up Databases

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
import logging
```
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.DescribeBackupDatabasesRequest()
    req.InstanceId = "cdb-7ghaiocc"
    req.StartTime = "2018-08-02 15:19:19"

    print(req)
    # Call the API you want to access through the client object. You need to pass in the request object
    resp = client.DescribeBackupDatabases(req)

    # A string return packet in JSON format is outputted
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    msg = traceback.format_exc()
    # Method 1
    print(msg)

DescribeBackupTables for Querying Backup Data Tables of the Specified Database

#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")
# Instantiate the client object to request the product (with TencentDB as an example)
client = cdb_client.CdbClient(cred, "ap-shanghai")

# Instantiate a request object: req = models.ModifyInstanceParamRequest()
req = models.DescribeBackupTablesRequest()
req.InstanceId = "cdb-7ghaiocc"
req.StartTime = "2018-08-02 15:19:19"
req.DatabaseName = "sissi"

# Call the API you want to access through the client object. You need to pass in the request object
resp = client.DescribeBackupTables(req)

# A string return packet in JSON format is outputted
print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)

---

### DescribeBackups for Querying Backup Logs

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.DescribeBackupsRequest()
    req.InstanceId = "cdb-7ghaiocc"

    # Call the API you want to access through the client object. You need to pass in the request object
    resp = client.DescribeBackups(req)
```
print resp

# A string return packet in JSON format is outputted
print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)

DescribeBinlogs for Querying Binary Logs

#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module

from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.DescribeBinlogsRequest()
    req.InstanceId = "cdb-7ghaiocc"

    # Call the API you want to access through the client object. You need to pass in the request object
    resp = client.DescribeBinlogs(req)

    # A string return packet in JSON format is outputted
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)

DescribeSlowLogs for Querying Slow Logs

#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module
```python
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
    req = models.DescribeSlowLogsRequest()
    req.InstanceId = "cdb-7ghaiocc"

    # Call the API you want to access through the client object. You need to pass in the request object
    resp = client.DescribeSlowLogs(req)

    # A string return packet in JSON format is outputted
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

**ModifyBackupConfig for Modifying the Database Backup Configuration**

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Import the TencentCloud API entry module

from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # Instantiate an authentication object. The Tencent Cloud account secretId and secretKey need to be passed in as the input parameters
    cred = credential.Credential("secretId", "secretKey")

    # Instantiate the client object to request the product (with TencentDB as an example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object: req = models.ModifyInstanceParamRequest()
```
req = models.ModifyBackupConfigRequest()
req.InstanceId = "cdb-ly6g3zj8"
req.ExpireDays = 10
req.StartTime = "06:00-10:00"
req.BackupMethod = "logical"
print(req)

# Call the API you want to access through the client object. You need to pass in the request object
resp = client.ModifyBackupConfig(req)

# A string return packet in JSON format is outputted
print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)