TencentDB for MySQL
Best Practice

Contents

Best Practice

Best Practices Overview

Last updated: 2023-09-04 11:04:10

This document describes the overview of TencentDB for MySQL best practices, which you can view based on your business needs.

Type	Link
Usage specifications	TencentDB for MySQL Usage Specifications
Access and connection	• How to Access TencentDB for MySQL from Light Cloud Servers via CCN • Configure automatic reconnection for applications • Creating a VPC for TencentDB for MySQL
Parameters	Impact of Modifying MySQL Primary Instance Parameters
Errors	Limitations of MyISAM Automatic Conversion to InnoDB Engine
Higher performance	• Enhancing Business Load Capacity with TencentDB for MySQL • Enhancing TencentDB for MySQL Performance with Read/Write Separation
Architecture building	• Two-Location, Three-Center Disaster Recovery Construction • Building an All-Scenario High-Availability Architecture
Building applications/websites	• Building LAMP Stack Web Applications • Building a Drupal Website
Calling APIs via Python	• Instance Purchase • Instance Management • Backup Task

Building All-Scenario High-Availability Architecture

Last updated: 2023-09-01 17:24:06

Based on cross-AZ deployment, TencentDB for MySQL provides remote backups and supporting tools, together with multiple features such as database proxy and elastic CPU expansion, to comprehensively guarantee the secure and stable operation of your business. This document describes how to build an all-scenario high-availability architecture through TencentDB for MySQL.

Example

Database plays a vital role in the core business of an enterprise, and data is the basic resource and lifeline of an enterprise. For this reason, a high-availability database architecture is necessary to ensure the stable production and operation of the enterprise. The enterprise will suffer a great deal in business running and economic loss if there are issues such as database downtime, data loss or unavailability. Therefore, TencentDB for MySQL launched the all-scenario high-availability architecture (AS-HAA) to ensure stable business operation in all aspects and processes.

Preparations

You have registered a Tencent Cloud account and completed identity verification.

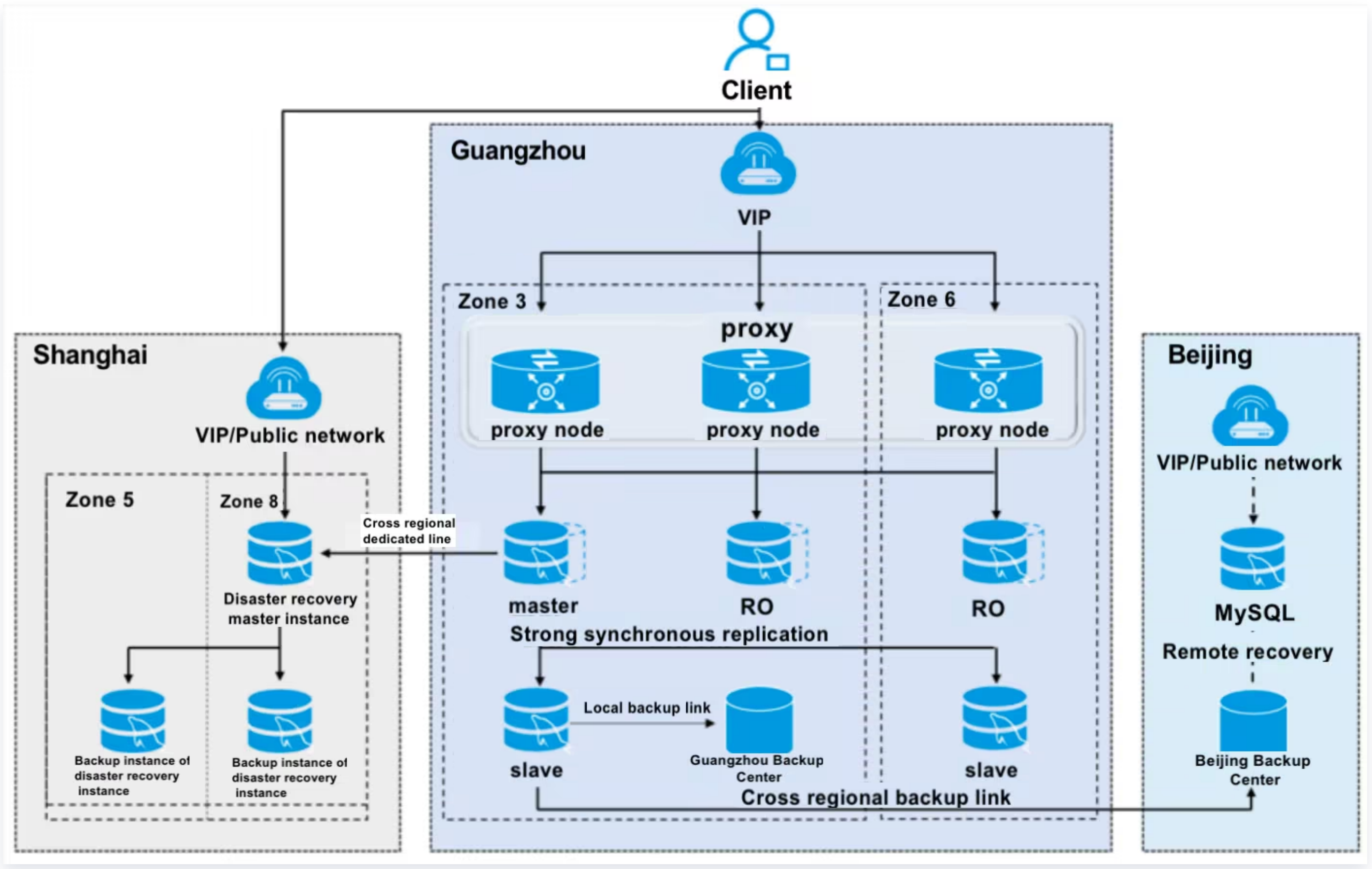
- Sign up for a Tencent Cloud account
- Complete identity verification

Advantages

- High stability: The capabilities like rapid expansion, load balancing, automatic elastic scaling, and nearby access are provided to ensure an environment with high scalability, smooth operation, and low network latency.
- High availability: All components can be deployed across AZs or regions, reducing the risk of single point of failure. Business downtime and data loss rates can be reduced through mechanisms such as backup and data sync.
- High data processing capacity and quick response speed: The database proxy distributes the load to multiple database nodes, thereby improving data processing capacity and response speed.
- High fault diagnosis and recovery efficiency: DBbrain monitors 7x24 hours to help DBAs quickly locate and solve problems; mechanisms such as backup and data sync are also

helpful for failure recovery and data recovery, shortening the time for bug fixes.

Instructions for Building AS-HAA



Step 1. Deploy a high-availability architecture

I. Creating a three-node architecture instance

1. Log in to the TencentDB for MySQL purchase page, complete **Basic Configuration** and **Instance Configuration** as needed, and click **Next: Set Network and Database**.
   Basic Configuration
   - **Billing Mode:** Monthly subscription and pay-as-you-go billing are supported.
   - **Region:** Select the region that you want your TencentDB for MySQL instance to be deployed in. We recommend that you select the same region as the CVM instance to be connected to. Tencent Cloud services in different regions cannot communicate with each other over the private network. The region cannot be modified after purchase.
   - **Database Version:** Currently, TencentDB for MySQL 5.5 (need to add to allowlist), 5.6, 5.7, and 8.0.
   - **Engine:** Select InnoDB or RocksDB.
   - **Architecture:** Select **Three-Node**.

  - **Disk Type**: TencentDB for MySQL supports local disk and cloud disk.
  - **AZ**: Select different AZs for the source AZ and replica AZ.

  Instance Configuration
  - **Filter**: You can quickly filter the needed CPU and memory specifications for the instance. By default, all CPU and memory specifications are selected.
  - **Type**: General or dedicated. For more information, see Resource Isolation Policy
  - **Instance Specs**: Select specifications as needed.
  - **Hard Disk**: The disk space is used to store the files required by MySQL execution. Select the size of the hard disk space.

2. Configure **Network and Others** and **Database Settings** and click **Next: Confirm the configuration info**.
  Network and Others
  - **Network**: VPC is supported. You can select the network and subnet for the instance.
  - **Custom Port**: The database access port, which is 3306 by default.
  - **Security Group**: For more information on security group creation and management, see TencentDB Security Group Management.

  > **Note**
  > Port 3306 must be opened for the TencentDB for MySQL instance through the inbound rule of the security group. The instance uses private network port 3306 by default and supports custom port. If the default port is changed, the new port should be opened in the security group.

  - **Project**: Select a project to which the database instance belongs. The default project will be used if you don't specify one.
  - **Tag**: It facilitates resource categorization and management.
  - **Alarm Policy**: You can create alarm policies to trigger alarms and send alarm notifications when the Tencent Cloud service status changes.

  Database Settings
  - **Instance Name**: Set a name for the instance now or later.
  - **Data Replication Mode**: Async, semi-sync, and strong sync replication modes are supported.
  - **Parameter Template**: Besides the system parameter template provided by TencentDB, you can create a custom parameter template. For more information, see Managing Parameter Template.
  - **Character Set**: LATIN1, GBK, UTF8, and UTF8MB4 are supported. The default character set is UTF8.

- **Collation**: The instance character set provides a case- and accent-sensitive collation for system data.
- **Table Name Case Sensitivity**: Whether the table name is case sensitive.

> **Note**
> In MySQL 8.0, the table name case sensitivity can be set only during instance purchase. For other versions, you can modify the `lower_case_table_names` parameter to set the case sensitivity for a table name. For detailed directions, see Setting Instance Parameters.

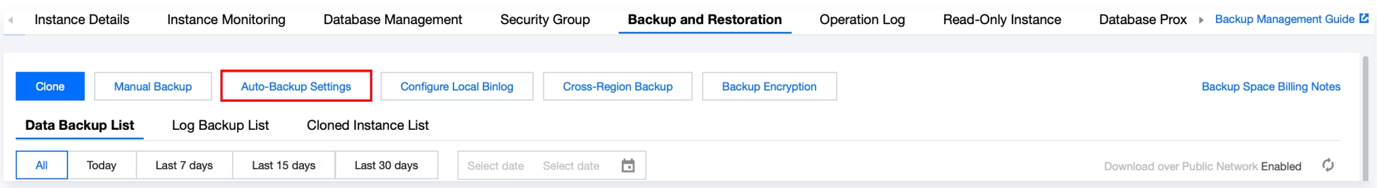
- **Password Complexity**: You can set the password complexity to improve the database security, which is disabled by default.
- **Root Password**: Set the password of the root account (the default user name for a new MySQL database is "root"). If you select **Set After Creation**, you can reset the password after creating the instance. For more information, see [Resetting Password](https://www.tencentcloud.com/document/product/236/31901)

3. Confirm the selected configuration items (if you need to modify them, click **Edit** to return to the corresponding step and make changes), read and indicate your consent to the **Terms of Service**, confirm the **Validity Period** and **Quantity**, and click **Buy Now**.
4. You will be returned to the instance list after you purchase the instance. The instance will be in the **Delivering** status. You can use the instance after around 3-5 minutes when its status changes to **Running**.

II. Setting local backup and cross-region backup|

Configure Local Backup

1. Log in to the TencentDB for MySQL console, click an **instance ID** on the instance list page to enter the management page, and select **Backup and Restoration* > Auto-Backup Settings**.

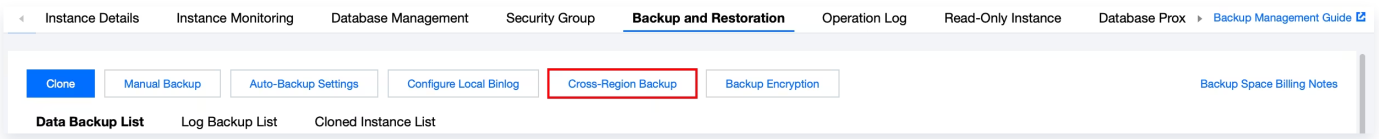


2. In the pop-up Backup Settings dialog box, select the desired backup parameters and click **OK**.

Enabling cross-region backup

1. Log in to the TencentDB for MySQL console. In the instance list, click an **instance ID** or **Manage** in the **Operation** column to enter the instance management page.

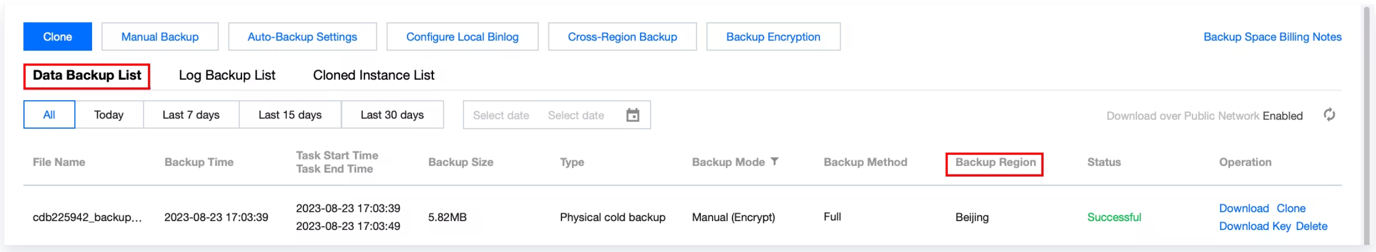
2. On the instance management page, select **Backup and Restoration** > **Cross-Region Backup**.



3. In the **Cross-Region Backup** window, complete the configuration and click **OK** to enable cross-region backup. Read and **select** the **Cross-Region Backup Space Billing Notes**, and click **OK**.

- Cross-Region Backup: It is disabled by default. You need to enable it here.
- Back up Binlog: When cross-region backup is enabled, it will be enabled automatically and can be disabled separately.
- Backup Region: Select one or two regions other than the source instance region.
- Backup Retention Period: 7 days by default. Value range: 3-1830 days. Backup sets will be deleted automatically upon expiration.

4. After cross-region backup is completed, the backup will be synced to the target region and can be queried in the backup list of the source instance.



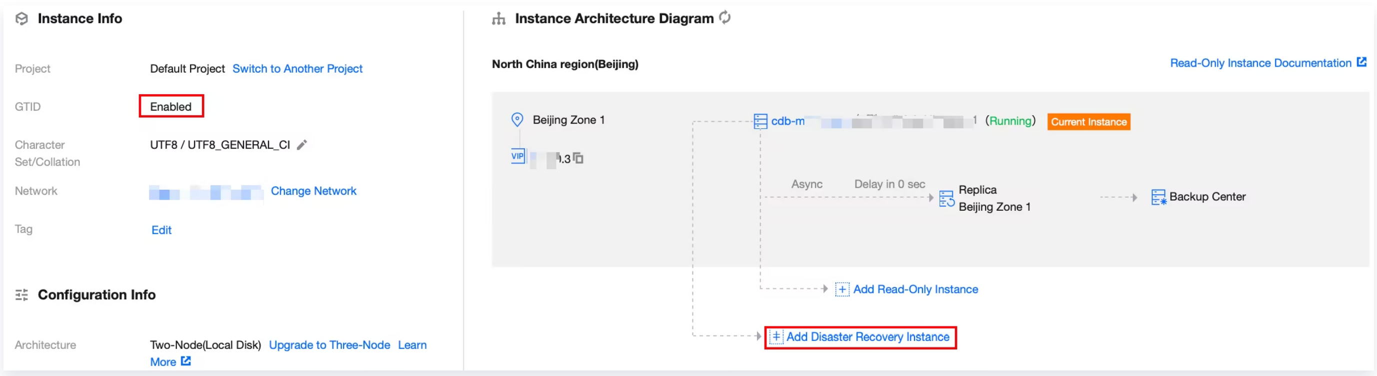
File Name	Backup Time	Task Start Time Task End Time	Backup Size	Type	Backup Mode	Backup Method	Backup Region	Status	Operation
cdb225942_backup...	2023-08-23 17:03:39	2023-08-23 17:03:39 2023-08-23 17:03:49	5.82MB	Physical cold backup	Manual (Encrypt)	Full	Beijing	Successful	Download Clone Download Key Delete

For cross-region backup files, all selected backup regions are displayed in the **Backup Region** column.

III. Creating a remote disaster recovery instance

1. Log in to the TencentDB for MySQL console. In the instance list, click an **instance ID** or **Manage** in the **Operation** column to enter the instance details page.
2. Make sure that the GTID feature is enabled by viewing the basic information of the instance on the **Instance Details** page. Click **Add Disaster Recovery Instance** in the instance

architecture diagram to enter the disaster recovery instance purchase page.



3. On the purchase page, set basic information of the disaster recovery instance such as **Billing Mode** and **Region**. For the region, select a region different from that of the three-node architecture instance created in step 1.

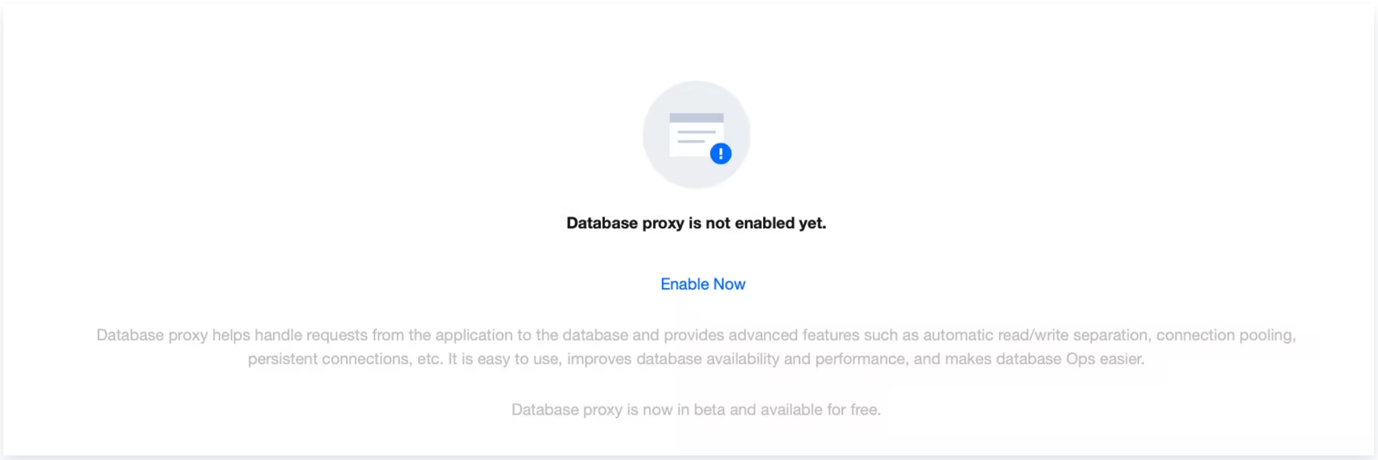
> **Note**
> - The time required to complete the creation depends on the amount of data, and no operations can be performed on the source instance in the console during the creation. We recommend you do so at an appropriate time.
> - If the sync policy is **Sync Now**, the data will be synced immediately when the disaster recovery instance is created.
> - Only the data of the entire instance can be synced. Make sure that the disk space is sufficient.
> - You need to ensure that the source instance is in the running state and none of configuration adjustment tasks, restart tasks, and other modification tasks are executed. Otherwise, the sync task may fail.

4. After confirming that everything is correct, click **Buy Now** and wait for the delivery of disaster recovery instance.
5. Return to the instance list. After the status of the instance changes to **Running**, it can be used normally.

Step 2. Enable the database proxy

1. Log in to the TencentDB for MySQL console. In the instance list, select the source instance for which to enable database proxy and click its **instance ID** or **Manage** in the **Operation** column to enter the instance management page.

2. On the instance management page, select the **Database Proxy** tab and click **Enable Now**.

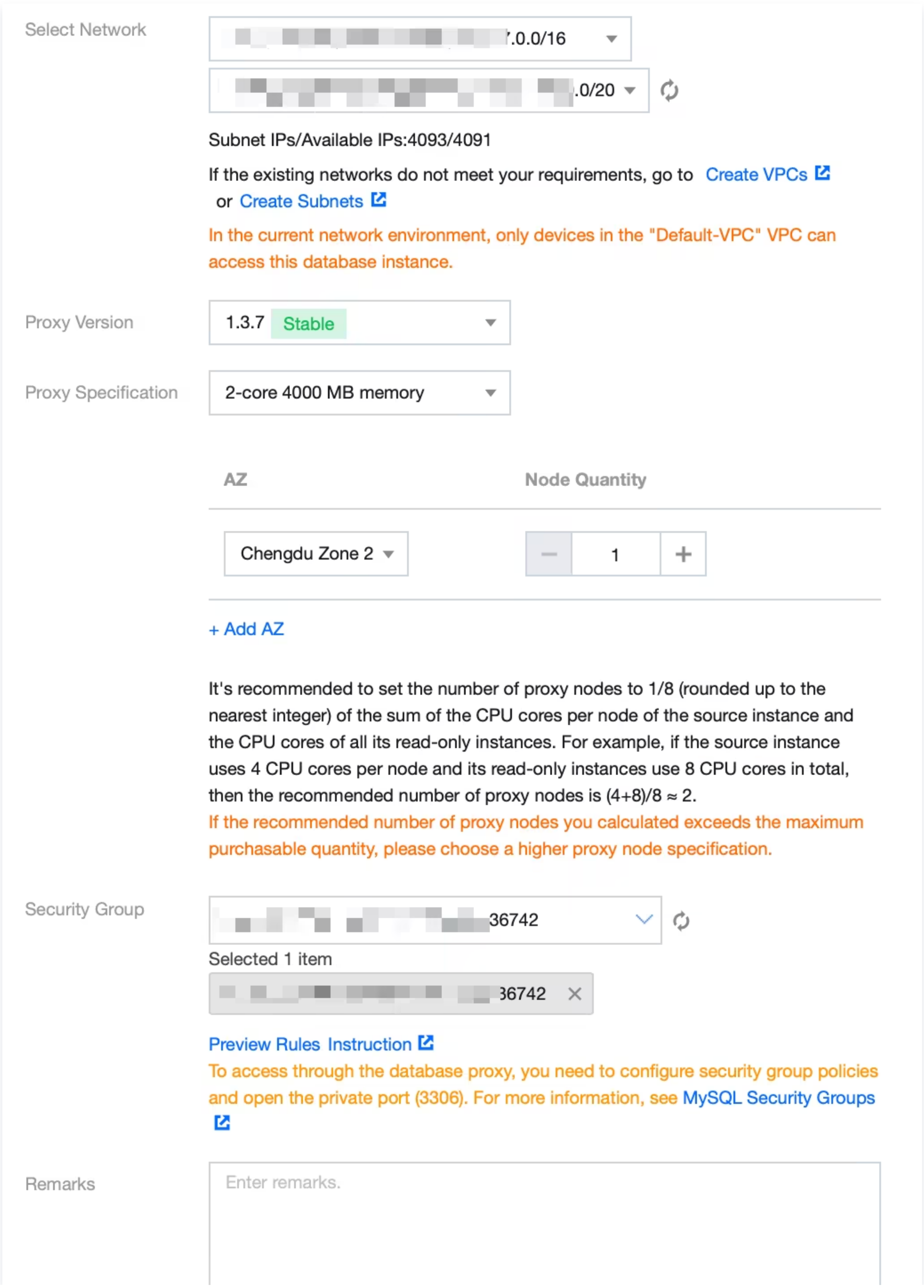

Database proxy is not enabled yet.

Enable Now

Database proxy helps handle requests from the application to the database and provides advanced features such as automatic read/write separation, connection pooling, persistent connections, etc. It is easy to use, improves database availability and performance, and makes database Ops easier.

Database proxy is now in beta and available for free.

3. In the pop-up window, configure the following items and click **OK**.



Select Network

[redacted].0.0/16

[redacted].0/20

Subnet IPs/Available IPs:4093/4091

If the existing networks do not meet your requirements, go to Create VPCs or Create Subnets

In the current network environment, only devices in the "Default-VPC" VPC can access this database instance.

Proxy Version: 1.3.7 Stable

Proxy Specification: 2-core 4000 MB memory

AZ	Node Quantity
Chengdu Zone 2	1

+ Add AZ

It's recommended to set the number of proxy nodes to 1/8 (rounded up to the nearest integer) of the sum of the CPU cores per node of the source instance and the CPU cores of all its read-only instances. For example, if the source instance uses 4 CPU cores per node and its read-only instances use 8 CPU cores in total, then the recommended number of proxy nodes is (4+8)/8 ≈ 2.

If the recommended number of proxy nodes you calculated exceeds the maximum purchasable quantity, please choose a higher proxy node specification.

Security Group

[redacted]36742

Selected 1 item

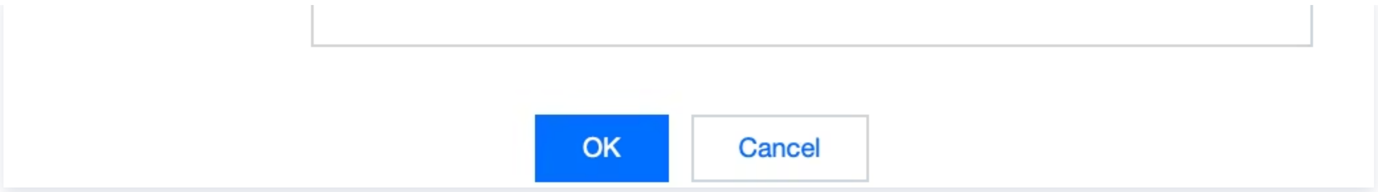
[redacted]36742

Preview Rules Instruction

To access through the database proxy, you need to configure security group policies and open the private port (3306). For more information, see MySQL Security Groups

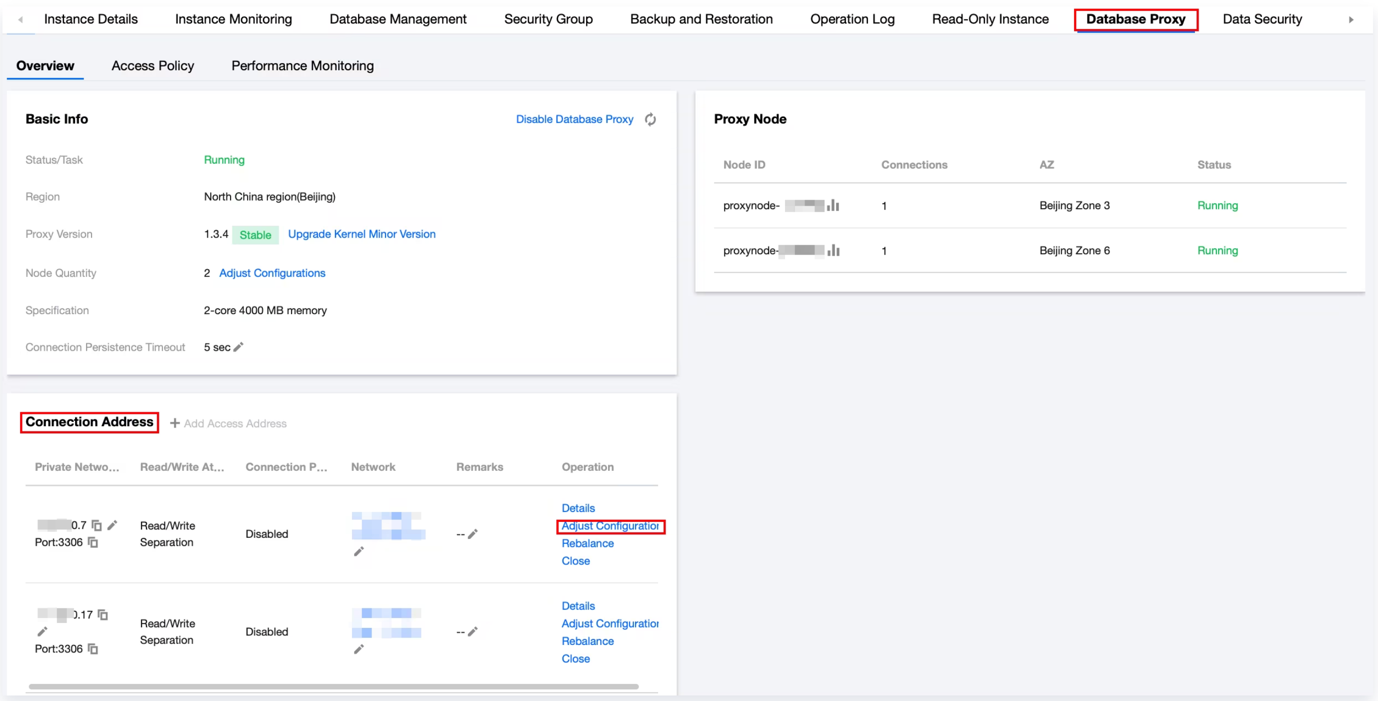
Remarks

Enter remarks.


OK
Cancel

Category	Note
Networking	Select the network of the database proxy, which can only be a VPC.
Proxy Specification	Select 2-core 4000 MB memory, 4-core 8000 MB memory, or 8-core 16000 MB memory.
AZ and Node Quantity	1. Select the AZ of the database proxy. You can make multiple selections by clicking **Add AZ**. The number of selectable AZs depends on how many AZs are available in the current region. You can select up to three AZs. 2. Select the number of nodes. We recommend that you set the quantity to 1/8 (rounded up) of the total number of CPU cores on the source and read-only instances; for example, if the source instance has 4 CPU cores, and the read-only instance has 8 CPU cores, then the recommended node quantity will be (4 + 8) / 8 ≈ 2. **Note:** If the selected proxy and the source instance are not in the same AZ and you access the instance through the proxy, the write performance may be reduced. If the recommended number of proxy nodes you calculated exceeds the maximum purchasable quantity, choose a higher proxy node specification.
Security Group	The security group of the source instance is selected by default. You can also select another existing security group or create a new one as needed. **Note:** To access through the database proxy, you need to configure security group policies and open the private port (3306). For more information, see TencentDB Security Group Management.
Remarks	(Optional) Enter the remarks of the database proxy service to be enabled.

4. After the enablement is successful, you can directly access the database proxy through the private access address under **Database Proxy** page > **Overview** > **Connection Address**, and you can configure access policy for the database proxy address. Then, find the target access address under **Database Proxy** > **Overview** > **Connection Address**, and click **Adjust Configurations** in the **Operation** column.



Basic Info

Status/Task	Running
Region	North China region(Beijing)
Proxy Version	1.3.4 Stable
Node Quantity	2
Specification	2-core 4000 MB memory
Connection Persistence Timeout	5 sec

Proxy Node

Node ID	Connections	AZ	Status
proxynode-	1	Beijing Zone 3	Running
proxynode-	1	Beijing Zone 6	Running

Connection Address

Private Netwo...	Read/Write At...	Connection P...	Network	Remarks	Operation
0.7 Port:3306	Read/Write Separation	Disabled		--	Details Adjust Configuration Rebalance Close
0.17 Port:3306	Read/Write Separation	Disabled		--	Details Adjust Configuration Rebalance Close

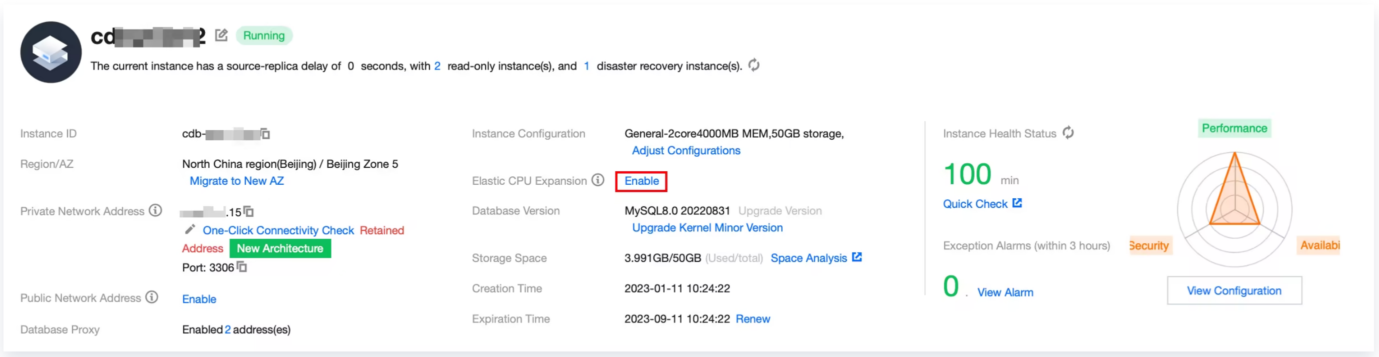
5. In the pop-up window, set the policy configuration and click **OK**.

Category	Note
Read-Write Attribute	Modify the read-write attribute of the proxy access address, which can be **Read/Write Separation** or **Read-Only**.
Remove Delayed RO Instances	Set the **Remove Delayed RO Instances** policy. After this option is enabled, you can set **Delay Threshold** and **Least RO Instances**. The system will try removing or restoring a failed read-only instance no matter whether this option is enabled. **Delay Threshold**: Enter an integer greater than or equal to 1 (in seconds). **Least RO Instances**: It is subject to the number of read-only instances owned by the source instance. If it is set to `0`, when all read-only nodes are removed, all read requests will be routed to the source instance until at least one of the removed read-only instances rejoins the database proxy to continue processing the read requests.
Connection Pool Status	The connection pool feature mainly mitigates the instance load caused by frequent new connections in non-persistent connection business scenarios. After this option is enabled, you can select the supported connection pool type, which currently can only be the session-level connection pool by default.
Transaction Split	You can set whether to enable this feature. After it is enabled, reads and writes in one transaction will be separated to different instances for execution, and read requests will be forwarded to read-only instances to reduce the load of the source instance.
Assign Read Weight	You can select **Assigned by system** or **Custom**. If multiple AZs are configured when the database proxy is enabled, you can separately assign the weights of proxy nodes in different AZs.
Failover (with **Read-Write Attribute** being **Read/Write Separation**)	You can set whether to enable this feature. After it is enabled, if database proxy fails, the database proxy address will route requests to the source instance.
Apply to Newly Added RO Instances	Enable or disable this parameter. After it is enabled, if you purchase new read-only instances, they will be automatically added to the database proxy. If **Assign Read Weight** is set to **Assigned by system**, newly purchased read-only instances will be assigned with the default weight based on their specification. If **Assign Read Weight** is set to **Custom**, when newly purchased

	read-only instances are added to the RO group, their weights will be 0 by default, which can be modified in the configuration of the database proxy's connection address.

Step 3: Enable CPU Elastic Expansion – Automatic Scaling

1. Log in to the TencentDB for MySQL console. In the instance list, click an **instance ID** or **Manage** in the **Operation** column to enter the instance details page.
2. Select **Instance Details** > **CPU Elastic Expansion**, and click **Enable**.

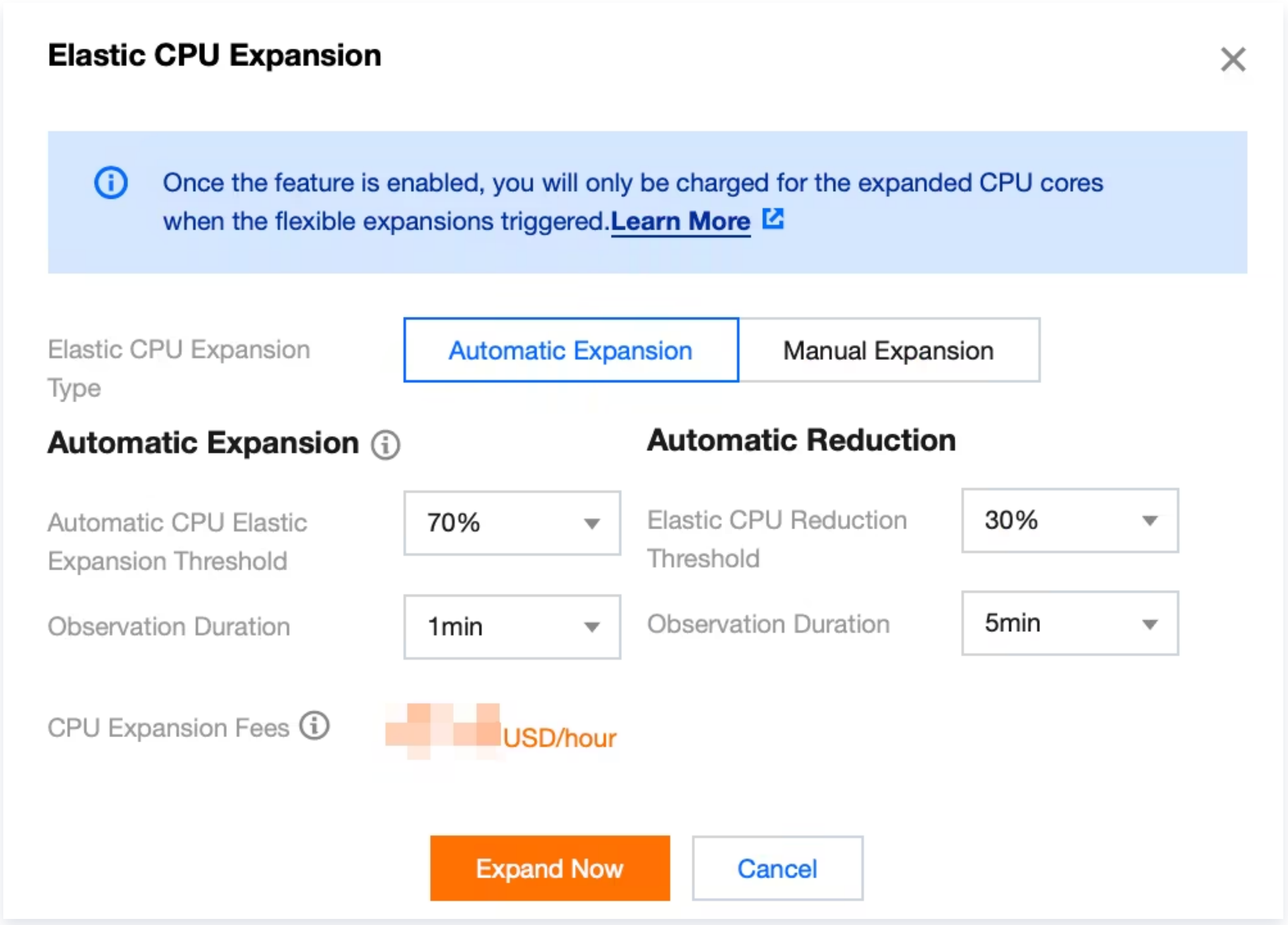


cd[redacted]2 Running

The current instance has a source-replica delay of 0 seconds, with 2 read-only instance(s), and 1 disaster recovery instance(s).

Instance ID	cdb-[redacted]	Instance Configuration	General-2core4000MB MEM,50GB storage, Adjust Configurations	Instance Health Status	100 min Quick Check
Region/AZ	North China region(Beijing) / Beijing Zone 5 Migrate to New AZ	Elastic CPU Expansion	Enable	Exception Alarms (within 3 hours)	0 . View Alarm
Private Network Address	[redacted].15 One-Click Connectivity Check Retained Address New Architecture Port: 3306	Database Version	MySQL8.0 20220831 Upgrade Version Upgrade Kernel Minor Version		
		Storage Space	3.991GB/50GB (Used/total) Space Analysis		
Public Network Address	Enable	Creation Time	2023-01-11 10:24:22		
Database Proxy	Enabled 2 address(es)	Expiration Time	2023-09-11 10:24:22 Renew		

3. In the **Elastic CPU Expansion** window, complete the following configurations, confirm the expansion fee, and click **Expand Now**.



Elastic CPU Expansion

Once the feature is enabled, you will only be charged for the expanded CPU cores when the flexible expansions triggered.Learn More

Elastic CPU Expansion Type: [✓ Automatic Expansion] Manual Expansion

Automatic Expansion

Automatic CPU Elastic Expansion Threshold	70%
Observation Duration	1min

Automatic Reduction

Elastic CPU Reduction Threshold	30%
Observation Duration	5min

CPU Expansion Fees [redacted]USD/hour

Category	Note
CPU Elastic Expansion Types	Select **Automatic Expansion**
Automatic CPU Elastic Expansion Threshold	Sets the threshold for automatic elastic expansion triggered by the average CPU utilization. Available options are 70%, 80%, and 90%.
Observation Duration	Sets the observation period. Available options are 1 minute, 3 minutes, 5 minutes, 10 minutes, 15 minutes, and 30 minutes. This parameter means that within the specified time period, the system will observe whether the average CPU utilization of the instance reaches the set expansion threshold. If yes, the system will trigger automatic elastic expansion.
Elastic CPU Reduction Threshold	Sets the threshold for automatic elastic reduction triggered by the average CPU utilization. Available options are 30%, 20%, and 10%.
Observation Duration	Sets the observation period. Available options are 5 minutes, 10 minutes, 15 minutes, and 30 minutes. This parameter means that within the specified time period, the system will observe whether the average CPU utilization of the instance reaches the set reduction threshold. If yes, the system will trigger automatic elastic reduction.

4. When the instance status/task changes from **Configuring elastic expansion policy···** to **Running**, automatic expansion is successfully enabled.

> **Note**
> After enabling automatic expansion, if you need to modify the elastic performance expansion policy, you can go to **Instance Details** > **CPU Elastic Expansion** and click **Modify** to reconfigure.

Instructions for Supporting Tools

DBbrain monitors and locates problems

You can use the DBbrain to realize database performance monitoring, security detection and optimization diagnosis. Through intelligent analysis and suggestions, it can help you quickly solve database performance and security problems and improve database efficiency.

Type	Performance Optimization Features	Description
Diagnostic analysis	Exception Diagnosis	The exception diagnosis feature provides you with real-time performance monitoring, health checks, and failure diagnosis and optimization, so that you can intuitively know the real-time operation status of database instances, locate newly appeared performance exceptions in real time, and optimize the system based on the optimization suggestions.
	Slow SQL Analysis	Offers SQL optimization suggestions after slow SQL analysis.
	Space Analysis	You can view the instance space utilization, including the sizes of data and logs, the daily increase in space utilization, the estimated number of available days, and the space used by the tables and databases of the instance and their change trends.
	SQL Optimization	The SQL optimization feature allows you to optimize SQL statements in just a few clicks and provides the corresponding execution plan interpretation and optimization advice.
	Audit log analysis	After analyzing the SQL performance, it provides optimization suggestions for poorly performing SQL statements by taking into account the index conditions and database/table design.
Locating and handling database session connection issues	Kill Session	Kill current session and kill sessions during a period.
	SQL Throttling	You can create SQL throttling tasks to control the database requests and SQL concurrency by setting the **SQL Type**, **Max Concurrency**, **Throttling Duration**, and **SQL Keyword**. Multiple tasks do not conflict with each other.
	Hotspot Update Protection	For businesses with frequent updates or flash sales, the hotspot update feature greatly optimizes the performance of the UPDATE operation on frequently updated rows.
		The autonomy service supports automatic SQL

Database autonomy	Autonomous Service	throttling and abnormal SQL killing. When the trigger conditions are met, the autonomous tasks of SQL throttling and abnormal SQL killing are automatically triggered to fix the database exceptions, with no human intervention required.

Chaotic Fault Generator simulates abnormal scenarios

Chaotic Fault Generator can help you simulate various emergencies and abnormal scenarios, and improve the fault tolerance and reliability of the system. After building a real database structure, the platform will conduct a full range of stress tests on the business, simulate traditional abnormal scenarios and perform automated deployment management, so as to better discover and diagnose potential problems in the system and improve business quality and reliability.
For detailed drill creation and process, see Getting Started of Chaotic Fault Generator .

Documentation

- Create a TencentDB for MySQL instance
- Backup Database
- Cross-Region Backup
- Create a DR instance
- Enable Database Proxy
- Transaction Splitting
- Anti-disconnection
- Configure database proxy read/write attributes
- Elastic CPU Expansion

Usage Specifications of TencentDB for MySQL

Last updated: 2023-09-01 17:26:38

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.

Privileges Management Specifications

- Note that the SUPER, SHUTDOWN, and FILE privileges are restricted in TencentDB for MySQL to ensure the stability and security. Therefore, the following error may occur when you execute SET statements:

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

  Solution: You can modify parameters on the **Database Management** > **Parameter Settings** tab in the console rather than run SET statements.
- Grant privileges on demand. It is sufficient to grant general applications only the DML privileges (SELECT, UPDATE, INSERT, DELETE).
- Grant privileges to users of general application at the database level.
- Allow authorized users to access TencentDB for MySQL only from specific IPs or IP ranges. 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.
- Separate management accounts from development accounts.

Operation Specifications

Supports and Limits

- Do not use weak passwords for enhanced instance security.
- 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 when connecting to or logging in to databases over the private network.

- Make sure that the client's MySQL version is the same as that of the TencentDB for MySQL instance when locally parsing the binlogs downloaded from the console; otherwise, garbled characters will be displayed during parsing. It is recommended to use mysqlbinlog v3.4 or higher.
- Enclose the URL with quotation marks when downloading cold backup files to a CVM instance over the private network in the console; otherwise, a 404 error will occur.

Suggestions

- Avoid performing online DDL operations during peak hours. You can use tools such as `pt-online-schema-change`.
- Avoid performing batch operations during peak hours.
- Avoid running an instance for multiple businesses to minimize the risk of mutual interference between businesses due to high coupling.
- 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`.
- Make an estimate of the resources required in advance and optimize the instances for promotional campaigns of your business. In case of a great demand for resources, contact your Tencent Cloud sales rep in a timely manner.

Database and Table Design Specifications

Supports and Limits

- Do not use MyISAM or MEMORY in MySQL v5.6 or higher as they are no longer supported. If MEMORY is required, you can use TencentDB for Redis or Memcached. If MyISAM databases are migrated to TencentDB for MySQL, MyISAM will be converted to InnoDB automatically.
- Create at least one index on the auto-increment column or create a composite index whose first column is the auto-increment column.
- Make sure that `row_format` is non-fixed.
- Make sure that each table has 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, primary key values are 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 `binlog_format` is row, deleting data in batches without the primary key can cause serious source-replica 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 monitoring metrics for TencentDB for MySQL in the Tencent Cloud Observability Platform (TCOP) console.
- Put the tables for the same type of businesses into one database when building databases 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.
- Use the DECIMAL type to store decimal values. The FLOAT and DOUBLE types have insufficient precision, especially for businesses involving money where the DECIMAL type must be used.
- Do not use the TEXT or BLOB type to store a large quantity 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. We recommend that you implement the foreign key logic at the application layer. Foreign key and cascade update are not suitable for high-concurrence scenarios, because they may reduce the insertion performance and lead to deadlock in case of high concurrence.
- To reduce the coupling between business logic and data storage, focus on data storage in the database and implement business logic primarily through the application layer. Minimize the use of advanced features such as stored procedures, triggers, functions, events, and views, as they have limited portability and scalability. If such objects exist in the instance, it is recommended not to set a definer by default to avoid migration failures caused by inconsistencies between the migration account and the definer.
- Do not use partitioned tables if you won't have a substantial business volume in the near future, because they are mainly used for archive management in the courier and

ecommerce industries. Do not rely on partitioned tables for performance enhancement, unless over 80% of queries in your business involve the partitioning key.
- Purchase read-only instances to implement read/write separation at the database level for business scenarios with a high read load and low requirement for consistency (data delay within seconds is acceptable).

Index Design Specifications

Supports and Limits

- Do not create indexes on the columns that are updated frequently and have a lower selectivity. Record updates will change the B+ tree, so creating indexes on frequently updated fields may greatly reduce the database performance.
- Put the column with the highest selectivity on the far left when creating a composite index; 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 a higher selectivity, 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, in `where a xxx and b = xxx` , b must be placed on the far left even if a has a higher selectivity, because a will not be used in the query.

Suggestions

- 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.
- Specify the index length when creating an index on the VARCHAR field. Do not index the entire column, because the VARCHAR column is often long, indexing the entire column will increase the maintenance costs, and specifying the index length can provide sufficient selectivity. You can use `count(distinct left(column name, index length))/count()` to check index selectivity.
- 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 a 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

Supports and Limits

- 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.
- When using `INSERT INTO t_xxx VALUES(xxx)`, explicitly specify the column attributes to be inserted to prevent data errors caused by changes in the table structure.
- Pay attention to the following 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 desclike`.
- 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.
- NOT, !=, NOT IN, etc. are used in queries.

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 source-replica 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 big 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 sql_no_cache id FROM test order by id LIMIT 80000,1 ) LIMIT 10 ;`.
- Avoid using a SQL statement with multi-level nested subqueries. The query optimizer prior to MySQL v5.5 can convert IN to EXISTS and does not go through the indexes. In this case, a large external table may result in poor performance.

Note

- It is difficult to completely avoid the aforementioned issues. The solution is to set the aforementioned conditions as secondary filtering conditions for indexes rather than as primary filtering conditions.
- If a large number of full-table scans are monitored, 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 is released. In daily Ops work, download slow query logs regularly for targeted optimization.

Configuring Automatic Application Reconnection

Last updated：2023-09-01 17:27:14

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

Example

In case of adjusting database instance specification or upgrading database engine, the source instance is overloaded and hanging, hardware fails, etc., the 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 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.

Note

- `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.

Impact of Modifying MySQL Source Instance Parameters

Last updated：2023-09-01 17:28:10

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`
Description: 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 source instance are modified, the parameters of the disaster recovery instance cannot be modified accordingly, as the source 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 source instance, then data sync will fail when the disaster recovery instance uses the corresponding logs, because the table name "TEst" already exists.

Note
To prevent issues caused by modifying this parameter, MySQL 8.0 only allows you to choose whether to enable table name case sensitivity during instance creation on the purchase page. For other versions, you can set this parameter both during instance creation on the purchase page and after purchase in the console.

auto_increment_increment

Default value: `1`
Description: 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 source instance are modified, those of the disaster recovery instance cannot be modified accordingly. When binlog_format is set as statement, only statement execution is recorded. In this case, if the increment column value of the source instance is modified but that of the disaster recovery instance is not modified accordingly, the data will be inconsistent between the source and slave instances.

auto_increment_offset

Default value: `1`
Description: It is used as the starting 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 source instance are modified, those of the disaster recovery instance cannot be modified accordingly. If the starting value of the auto-increment column in the source instance is modified but that of the disaster recovery instance is not modified accordingly, the data will be inconsistent between the source and slave instances.

sql_mode

Default value: `NO_ENGINE_SUBSTITUTION`
Description: 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 reported; 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`, and `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 are subject to the status of the strict mode.
- `ERROR_FOR_DIVISION_BY_ZERO` means that in strict SQL mode, if data is divided by zero during the INSERT or UPDATE process, an error rather than a warning will be generated, while in non-strict SQL mode, NULL will be returned.
- `NO_AUTO_CREATE_USER` prohibits the GRANT statement from creating a user with an empty password.
- `NO_ENGINE_SUBSTITUTION` means that if the storage engine is disabled or not compiled, an error will be reported.

Impact: After the parameters of the source instance are modified, those of the disaster recovery instance cannot be modified accordingly. If the source instance modifies the SQL mode and the disaster recovery instance does not synchronize the change, issues may arise when the SQL mode restrictions of the source instance are less strict than those of the disaster recovery instance. This can result in errors when executing SQL statements that were successful on the source instance, leading to data inconsistency between the source and disaster recovery instances.

Limits on Automatic Conversion from MyISAM to InnoDB

Last updated：2023-09-01 17:28:28

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

Example

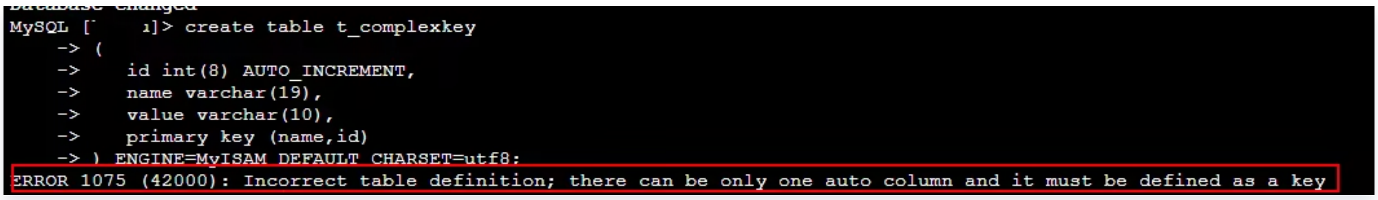
TencentDB for MySQL supports the InnoDB storage engine by default and no longer supports the MyISAM and Memory engines in MySQL 5.6 and later versions. For more information, please refer to Database Storage Engines.
When migrating or upgrading to TencentDB for MySQL 5.6 or later, the system automatically converts the MyISAM engine to the InnoDB engine.
Since the MyISAM engine supports composite primary keys with auto-increment columns, while the InnoDB engine does not, an error occurs when creating a table after converting from MyISAM to InnoDB. The error message is `ERROR 1075 (42000): Incorrect table definition; there can be only one auto column and it must be defined as a key.`
We recommend implementing the InnoDB engine's composite primary key with auto-increment column syntax by creating an index for the auto-increment column.

Solution

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

```
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:

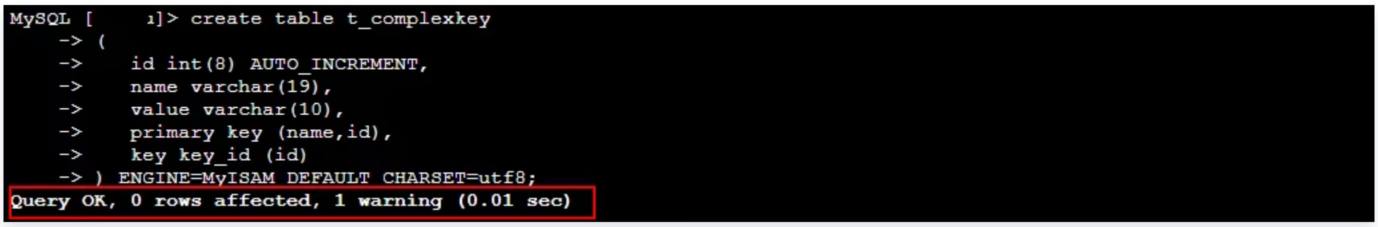


```
MySQL [[redacted]]> create table t_complexkey
    -> (
    ->    id int(8) AUTO_INCREMENT,
    ->    name varchar(19),
    ->    value varchar(10),
    ->    primary key (name,id)
    -> ) ENGINE=MyISAM DEFAULT CHARSET=utf8;
ERROR 1075 (42000): Incorrect table definition; there can be only one auto column and it must be defined as a key
```

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

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

The table is created successfully as follows:

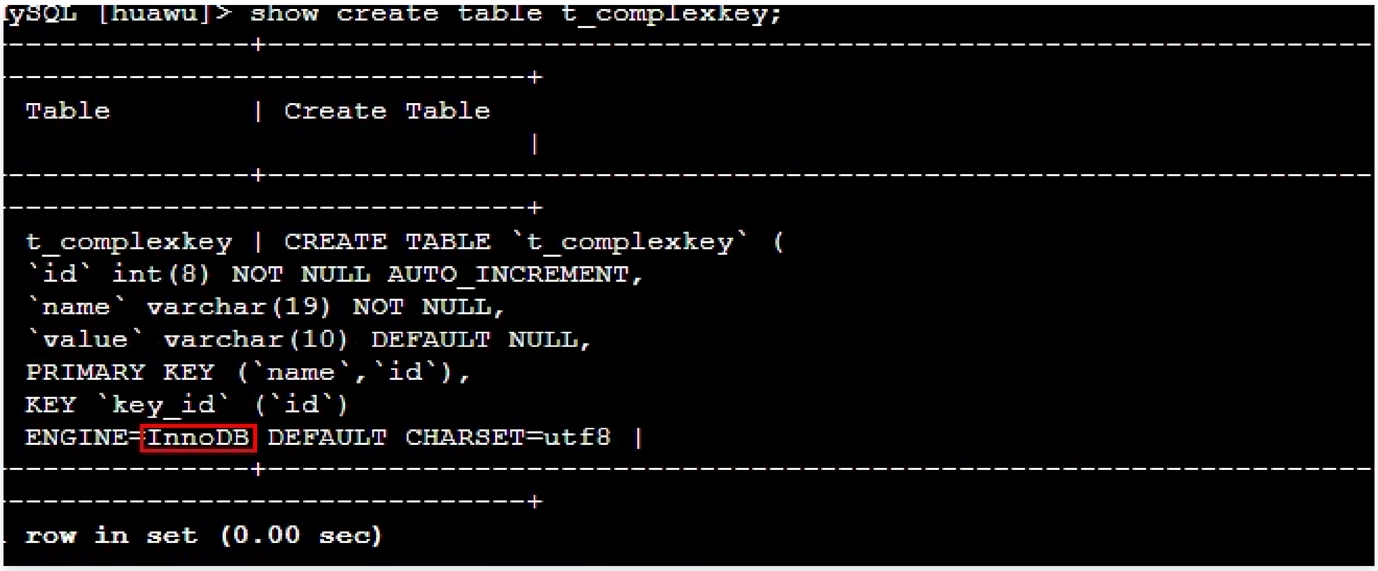


```
MySQL [[redacted]ı]> create table t_complexkey
    -> (
    ->     id int(8) AUTO_INCREMENT,
    ->     name varchar(19),
    ->     value varchar(10),
    ->     primary key (name,id),
    ->     key key_id (id)
    -> ) ENGINE=MyISAM DEFAULT CHARSET=utf8;
Query OK, 0 rows affected, 1 warning (0.01 sec)
```

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

```
show create table t_complexkey;
```

See below:



```
ySQL [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)
```

Creating VPCs for TencentDB for MySQL

Last updated: 2023-09-01 17:28:52

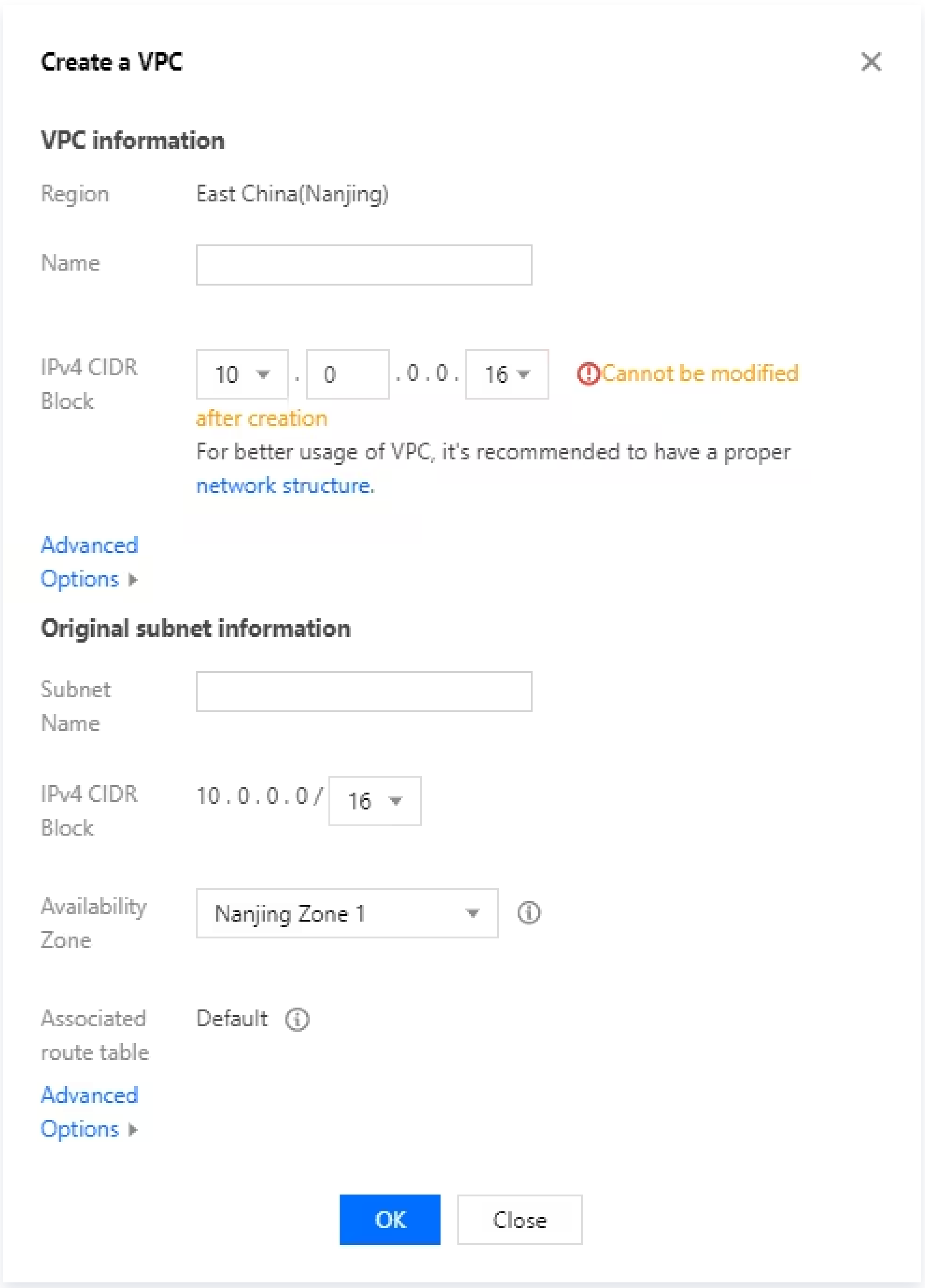
Tencent Cloud provides Virtual Private Cloud (VPC), a platform for hosting TencentDB instances. You can launch Tencent Cloud resources in a VPC, such as TencentDB instances. A common scheme is to share data between a TencentDB instance and a web server running in the same VPC. This document uses this scheme to create a VPC and add a TencentDB instance to it.
Below 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 the Virtual Private Cloud Console.
2. Select the region of the VPC at the top of the list and click **Create**.
3. Enter the VPC information and initial 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` (subnet mask range must be between 12 and 28)
    - `172.16.0.0` - `172.31.255.255` (mask range must be between 12 and 28)
    - `192.168.0.0` - `192.168.255.255` (mask range must be between 16 and 28)
- The subnet CIDR must be within or the same as the VPC CIDR.
  For example, if the VPC CIDR is `192.168.0.0/16`, the subnet CIDR within the VPC can be `192.168.0.0/16`, `192.168.0.0/17`, etc.

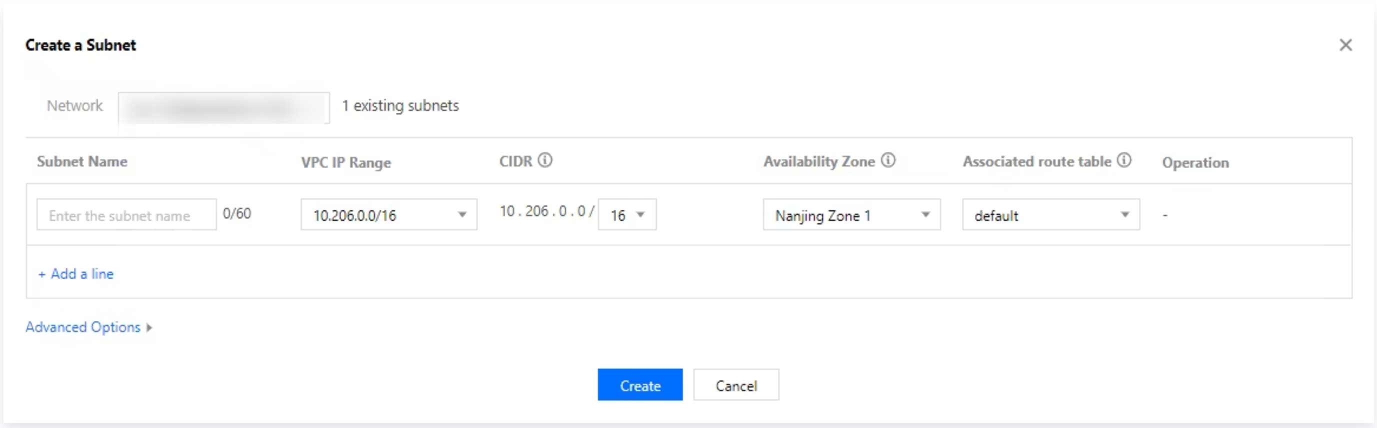


Step 2. Create a subnet

You can create one or more subnets at a time.

1. Log in to the Virtual Private Cloud 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 **Create**.

4. Enter the subnet’s name, CIDR, availability zone, and associated route table.


Create a Subnet

Network [redacted] 1 existing subnets

Subnet Name	VPC IP Range	CIDR	Availability Zone	Associated route table	Operation
Enter the subnet name 0/60	10.206.0.0/16	10 . 206 . 0 . 0 / 16	Nanjing Zone 1	default	-

+ Add a line

Advanced Options

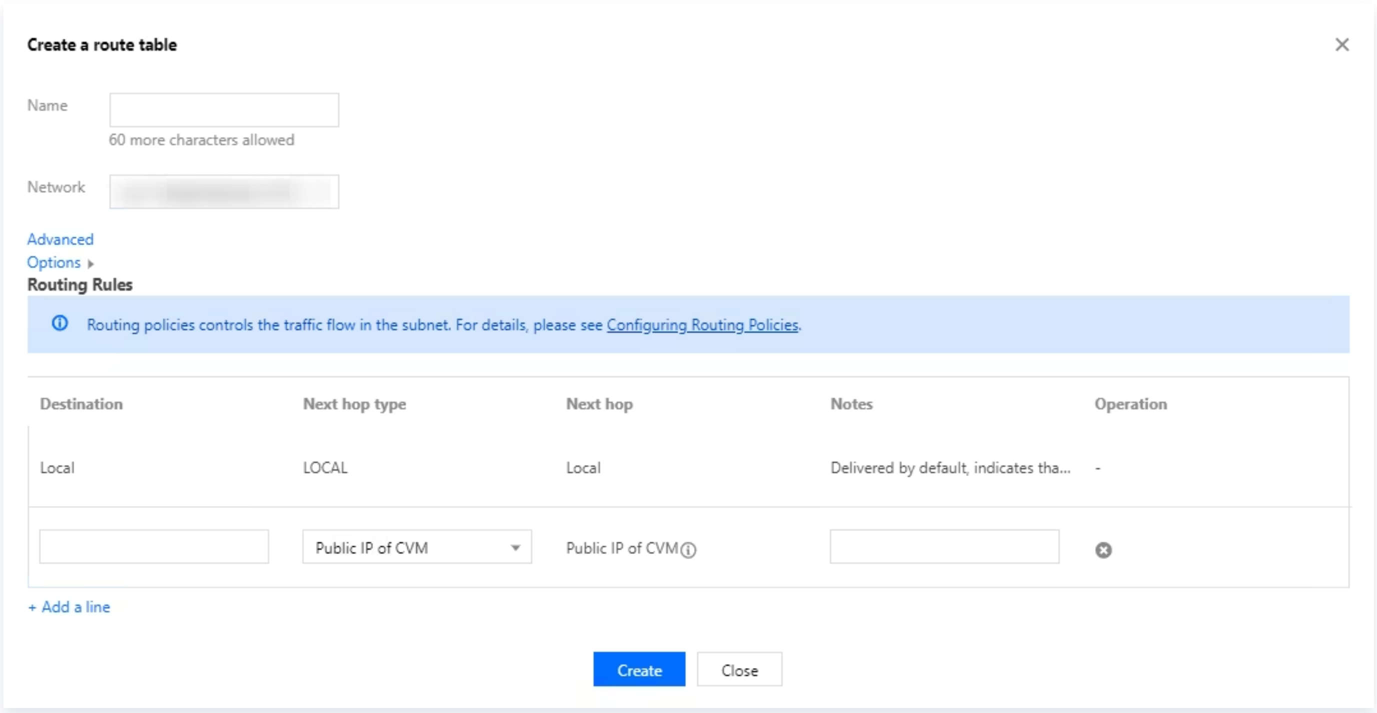
Create Cancel

5. (Optional) Click **+New line** to create multiple subnets at a time.
6. Click **Create**.

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 **Create**.
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.


Create a route table

Name

60 more characters allowed

Network [redacted]

Advanced Options

Routing Rules

Routing policies controls the traffic flow in the subnet. For details, please see Configuring Routing Policies.

Destination	Next hop type	Next hop	Notes	Operation
Local	LOCAL	Local	Delivered by default, indicates tha...	-
	Public IP of CVM	Public IP of CVM		

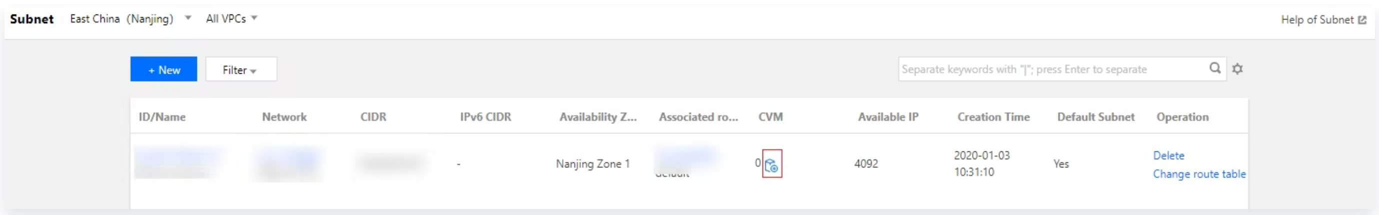
+ Add a line

Create Close

4. Click **Subnet** on the left sidebar, select the subnet to be associated with the route table, and click **More** > **Change route table** in the **Operation** column to associate it.

Step 4. Add a CVM instance

1. Log in to the Virtual Private Cloud 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.



Subnet East China (Nanjing) All VPCs Help of Subnet

+ New Filter Separate keywords with "|"; press Enter to separate

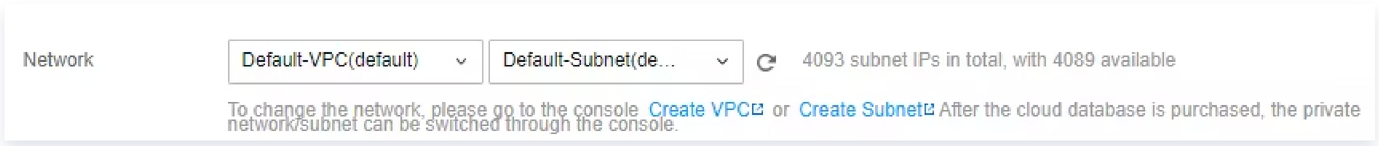
ID/Name	Network	CIDR	IPv6 CIDR	Availability Z...	Associated ro...	CVM	Available IP	Creation Time	Default Subnet	Operation
[redacted]	[redacted]	[redacted]	-	Nanjing Zone 1	[redacted] default	0	4092	2020-01-03 10:31:10	Yes	Delete Change route table

4. Follow the on-screen instructions to complete the purchase of a CVM instance. For more information, refer to the CVM documentation Purchase Methods.

Step 5. Add a TencentDB instance

Create a 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.



Network Default-VPC(default) Default-Subnet(de... 4093 subnet IPs in total, with 4089 available

To change the network, please go to the console Create VPC or Create Subnet After the cloud database is purchased, the private network/subnet can be switched through the console.

Existing database

1. In the Instance List, click an **instance ID** 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.

Enhancing Business Load Capacity with TencentDB for MySQL

Last updated：2023-09-01 17:29:33

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, v5.7, and v8.0, all of which are fully compatible with native MySQL. We recommend that you choose v5.6 and later, as they use more stable database kernels, deliver better system performance by optimizing the design of v5.5 and earlier, and come with a lot of appealing new features.
This document takes MySQL 5.7 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 follows:

- **Native JSON support**
  MySQLIn MySQL 5.7, 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 5.7, 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-related 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 tablespace online, thus eliminating one of the most common reasons for large shared tablespace files (ibdata1). In addition, MySQL 5.7 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 5.7, 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 5.7, see MySQL official documentation.

1.2 Select 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 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. We recommend that you 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 Performance Overview.

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 concurrency or sensitivity to read latency, it is recommended not to choose a small memory specification to ensure optimal database performance.

1.3 Select a disk

The disk space of a TencentDB for MySQL instance contains data, system, binlog, and temporary files. When the amount of written data exceeds the instance's disk capacity, if the instance is not upgraded, instance lock may be triggered. Therefore, when you purchase a disk, we recommend you 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 source-replica architecture. Source-replica 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 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 Connecting to MySQL Instance.

2. Take Read-Only Instances 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 enhance the ability to handle read load.
TencentDB for MySQL read-only instances are ideal for such issues. For more information, see Creating Read-Only Instance.
Read-only instances can also be used for read-only access in various businesses; for example, the source 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. TencentDB Disaster Recovery Solutions

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 Tencent Cloud private network to sync data and the replication is optimized at the level of MySQL kernel, 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 Nearby Access

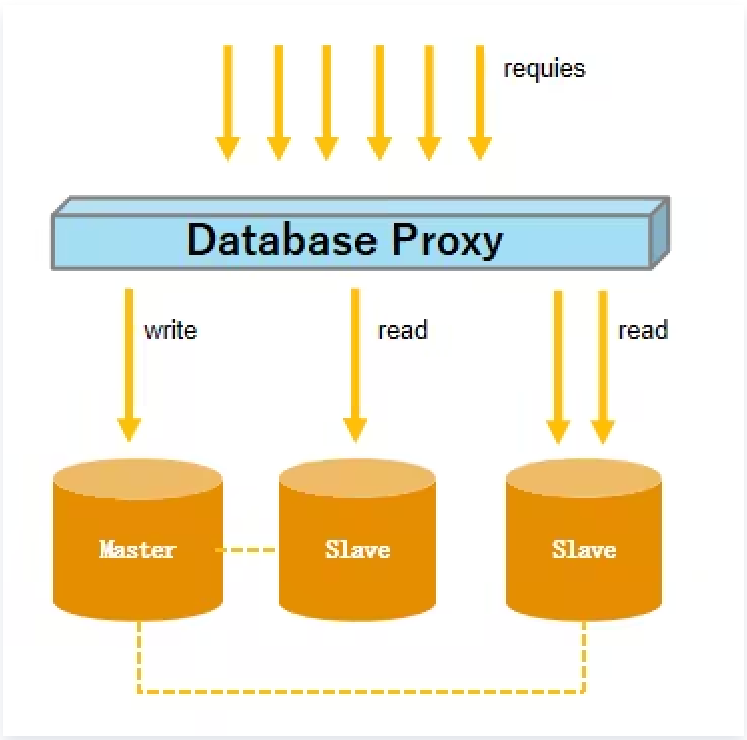
A disaster recovery instance also adopts the high-availability source-replica architecture. In addition, it can be accessed in a read-only manner, which helps enable local access to your businesses for end users in different regions.

Improving TencentDB for MySQL Performance with Read/Write Separation

Last updated: 2023-09-01 17:32:31

This document describes how to enable the read/write separation feature through the database proxy service to achieve horizontal scaling and improve the performance of TencentDB for MySQL.

Implementing Read/Write Separation Architecture via Database Proxy



Database Proxy Overview

Database proxy is a network proxy service between the TencentDB service and the application service. It is used to proxy all requests when the application service accesses the database.
The database proxy access address is independent of the original database access address. Requests arriving at the proxy address are all relayed through the proxy cluster to access the source and replica nodes of the database. Read/Write separation is implemented, so that read requests are forwarded to read-only instances, which lowers the load of the source database.

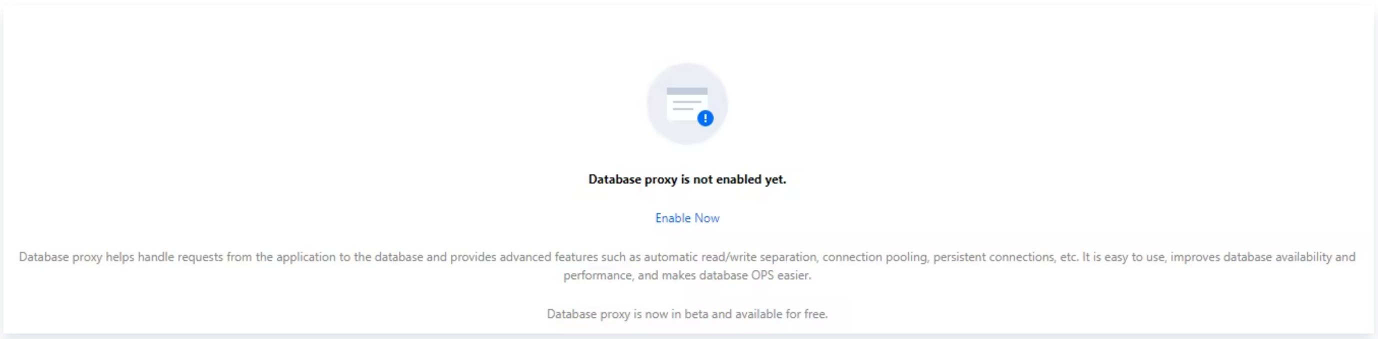
Automatic read/write separation

Currently, your business may face scenarios such as more reads and less writes as well as unpredictable business loads. In application scenarios with a large number of read requests, a single instance may not be able to withstand the load of read requests, potentially affecting the business.
To implement the auto scaling of read capabilities and mitigate the pressure on the database, you can create one or multiple read-only instances and use them to sustain high numbers of database reads. However, this solution requires that businesses can be transformed to support read/write separation, and the code robustness determines the quality of business read/write separation, which imposes high technical requirements and has low flexibility and scalability.
After creating a read-only instance, you can purchase the database proxy service to enable the read/write separation feature. Then, you can configure the database proxy address in your application so as to automatically forward write requests to the source instance and read requests to the read-only instance.

nabling Read/Write Separation via Database Proxy

Step 1. Enable the database proxy

1. Log in to the TencentDB for MySQL console. In the instance list, select the source instance for which to enable database proxy and click its ID or **Manage** in the **Operation** column to enter the instance management page.
2. On the instance management page, select the **Database Proxy** tab and click **Enable Now**.

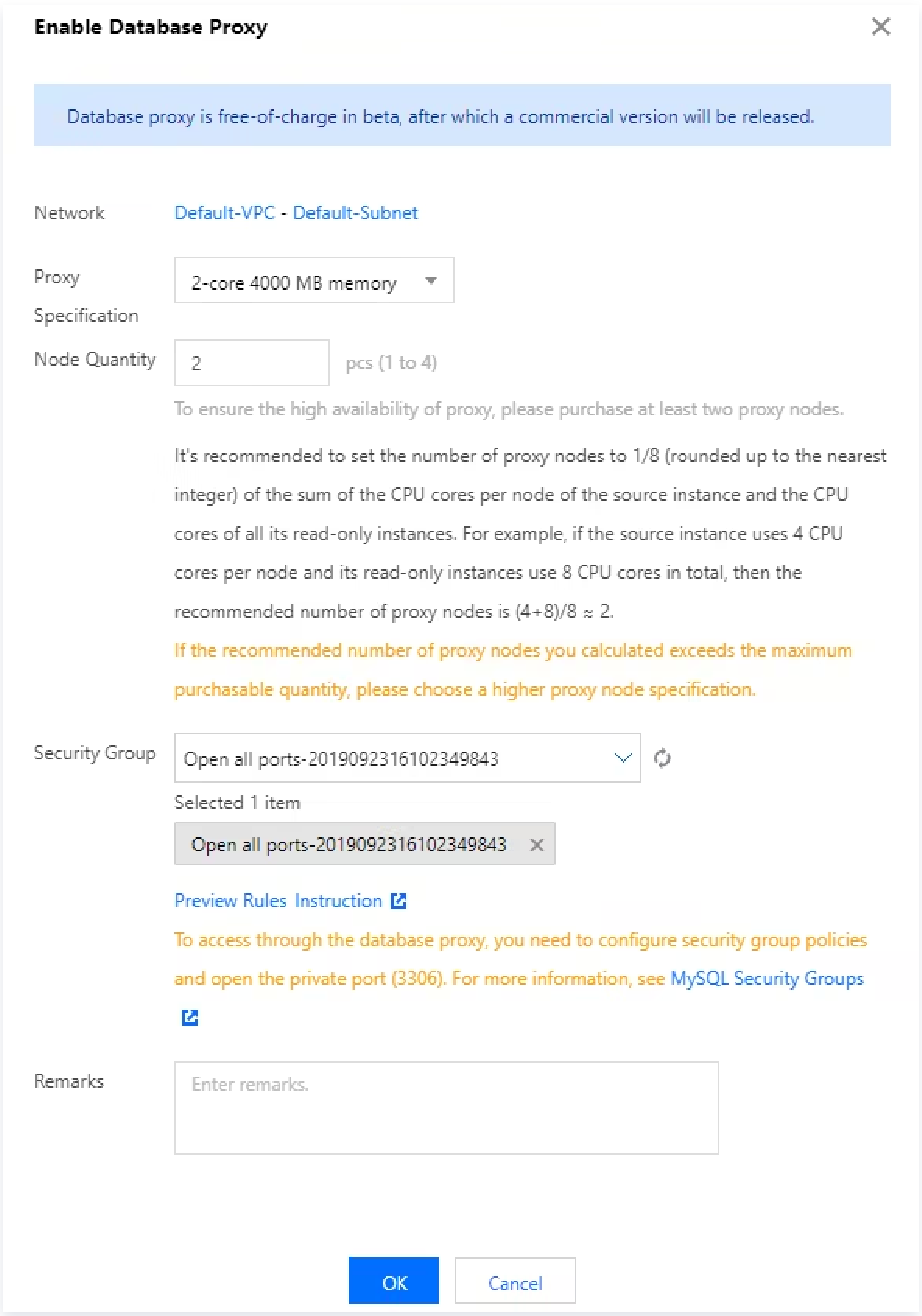

Database proxy is not enabled yet.

Enable Now

Database proxy helps handle requests from the application to the database and provides advanced features such as automatic read/write separation, connection pooling, persistent connections, etc. It is easy to use, improves database availability and performance, and makes database OPS easier.

Database proxy is now in beta and available for free.

3. In the pop-up window, configure the following items and click **OK**.



Enable Database Proxy

Database proxy is free-of-charge in beta, after which a commercial version will be released.

Network: Default-VPC - Default-Subnet

Proxy Specification: 2-core 4000 MB memory

Node Quantity: 2 pcs (1 to 4)

To ensure the high availability of proxy, please purchase at least two proxy nodes.

It's recommended to set the number of proxy nodes to 1/8 (rounded up to the nearest integer) of the sum of the CPU cores per node of the source instance and the CPU cores of all its read-only instances. For example, if the source instance uses 4 CPU cores per node and its read-only instances use 8 CPU cores in total, then the recommended number of proxy nodes is (4+8)/8 ≈ 2.

If the recommended number of proxy nodes you calculated exceeds the maximum purchasable quantity, please choose a higher proxy node specification.

Security Group: Open all ports-2019092316102349843

Selected 1 item

Open all ports-2019092316102349843

Preview Rules Instruction

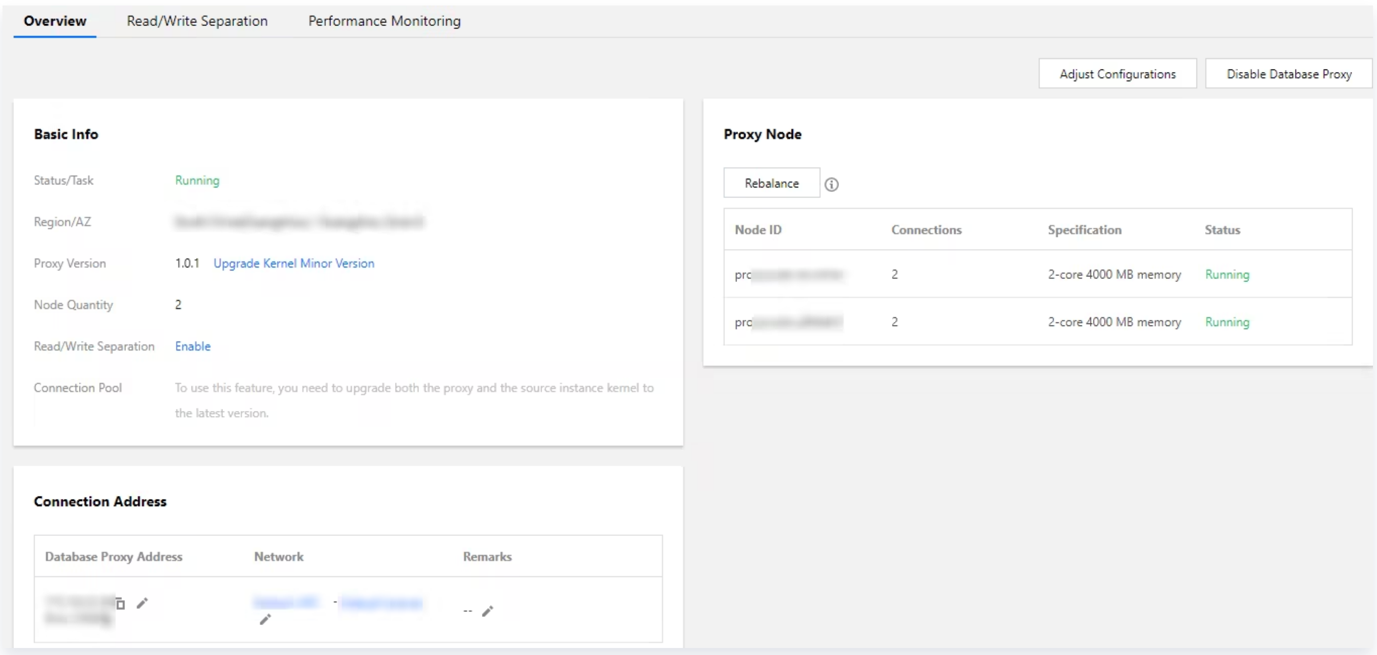
To access through the database proxy, you need to configure security group policies and open the private port (3306). For more information, see MySQL Security Groups

Remarks: Enter remarks.

Category	Note
Networking	Select the network of the database proxy, which can only be a VPC.
Proxy Specification	Select 2-core 4000 MB memory, 4-core 8000 MB memory, or 8-core 16000 MB memory.
AZ and Node Quantity	1. Select the AZ of the database proxy. You can make multiple selections by clicking **Add AZ**. The number of selectable AZs depends on how many AZs are available in the current region. You can select up to three AZs. 2. Select the number of nodes. We recommend that you set the quantity to 1/8 (rounded up) of the total number of CPU cores on the source and read-only instances; for example, if the source instance has 4 CPU cores, and the read-only instance has 8 CPU cores, then the recommended node quantity will be (4 + 8) / 8 ≈ 2. **Note:** 1. If the selected proxy and the source instance are not in the same AZ and you access the instance through the proxy, the write performance may be reduced. 2. If the recommended number of proxy nodes you calculated exceeds the maximum purchasable quantity, choose a higher proxy node specification.
Security Group	The security group of the source instance is selected by default. You can also select another existing security group or create a new one as needed. **Note:** To access through the database proxy, you need to configure security group policies and open the private port (3306). For more information, see TencentDB Security Group Management.
Remarks	(Optional) Enter the remarks of the database proxy service to be enabled.

4. After successfully enabling the service, you can manage proxy nodes and view their basic information on the database proxy page. You can also modify the access address, network type, and remarks of the database proxy, view and adjust the connection configuration, and perform rebalance in the **Connection Address** section.

> **Note**
> - You can view **Connections** in the proxy node list or view the performance monitoring data of each proxy node to check whether the numbers of connections on the nodes are unbalanced, and if so, you can distribute the connections by clicking **Rebalance**.
> - Rebalance will cause proxy nodes to restart, and the service will become unavailable momentarily during the restart. We recommend you restart the service during off-peak hours. Please make sure that your business has a reconnection mechanism.



Basic Info

Status/Task	Running
Region/AZ	[illegible]
Proxy Version	1.0.1 Upgrade Kernel Minor Version
Node Quantity	2
Read/Write Separation	Enable
Connection Pool	To use this feature, you need to upgrade both the proxy and the source instance kernel to the latest version.

Proxy Node

Node ID	Connections	Specification	Status
pro[illegible]	2	2-core 4000 MB memory	Running
pro[illegible]	2	2-core 4000 MB memory	Running

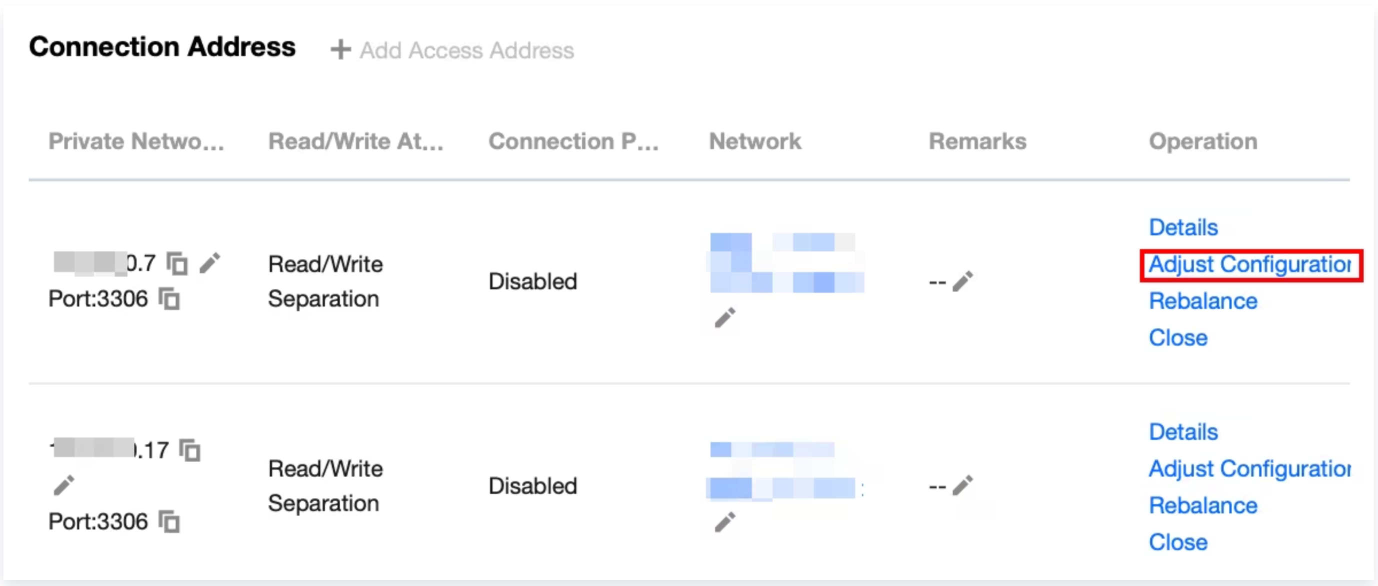
Connection Address

Database Proxy Address	Network	Remarks
[illegible]	[illegible] - [illegible]	--

Step 2. Enable database proxy read/write separation

1. Log in to the TencentDB for MySQL console. Select a region on the top, and click the target **instance ID** to enter the instance management page.
2. On the instance management page, select **Database Proxy** > **Overview**. Locate the target access address under Connection Address, and click **Adjust Configurations** in the

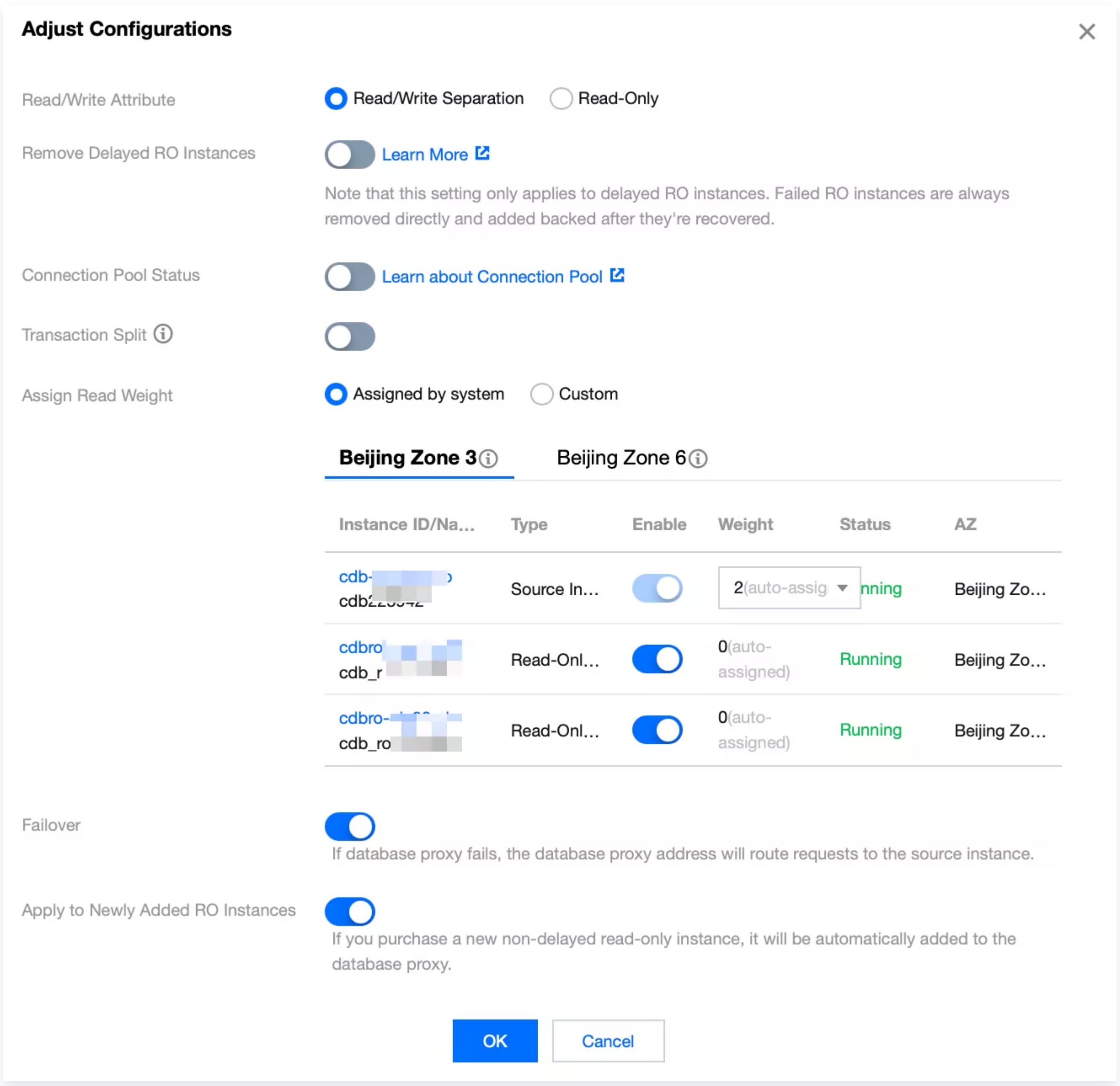
Operation column.



Connection Address + Add Access Address

Private Netwo...	Read/Write At...	Connection P...	Network	Remarks	Operation
0.7 Port:3306	Read/Write Separation	Disabled		--	Details Adjust Configuratio Rebalance Close
.17 Port:3306	Read/Write Separation	Disabled		--	Details Adjust Configuratio Rebalance Close

3. In the pop-up window, set **Read-Write Attribute**, assign the read weight, and click **OK**.



Adjust Configurations

Read/Write Attribute: [✓ Read/Write Separation] Read-Only

Remove Delayed RO Instances: Learn More

Note that this setting only applies to delayed RO instances. Failed RO instances are always removed directly and added backed after they're recovered.

Connection Pool Status: Learn about Connection Pool

Transaction Split

Assign Read Weight: [✓ Assigned by system] Custom

Beijing Zone 3 | Beijing Zone 6

Instance ID/Na...	Type	Enable	Weight	Status	AZ
cdb-[redacted] cdb[redacted]	Source In...	✓	2(auto-assig	nning	Beijing Zo...
cdbro[redacted] cdb_r[redacted]	Read-Onl...	✓	0(auto-assigned)	Running	Beijing Zo...
cdbro-[redacted] cdb_ro[redacted]	Read-Onl...	✓	0(auto-assigned)	Running	Beijing Zo...

Failover: ✓

If database proxy fails, the database proxy address will route requests to the source instance.

Apply to Newly Added RO Instances: ✓

If you purchase a new non-delayed read-only instance, it will be automatically added to the database proxy.

OK Cancel

Note

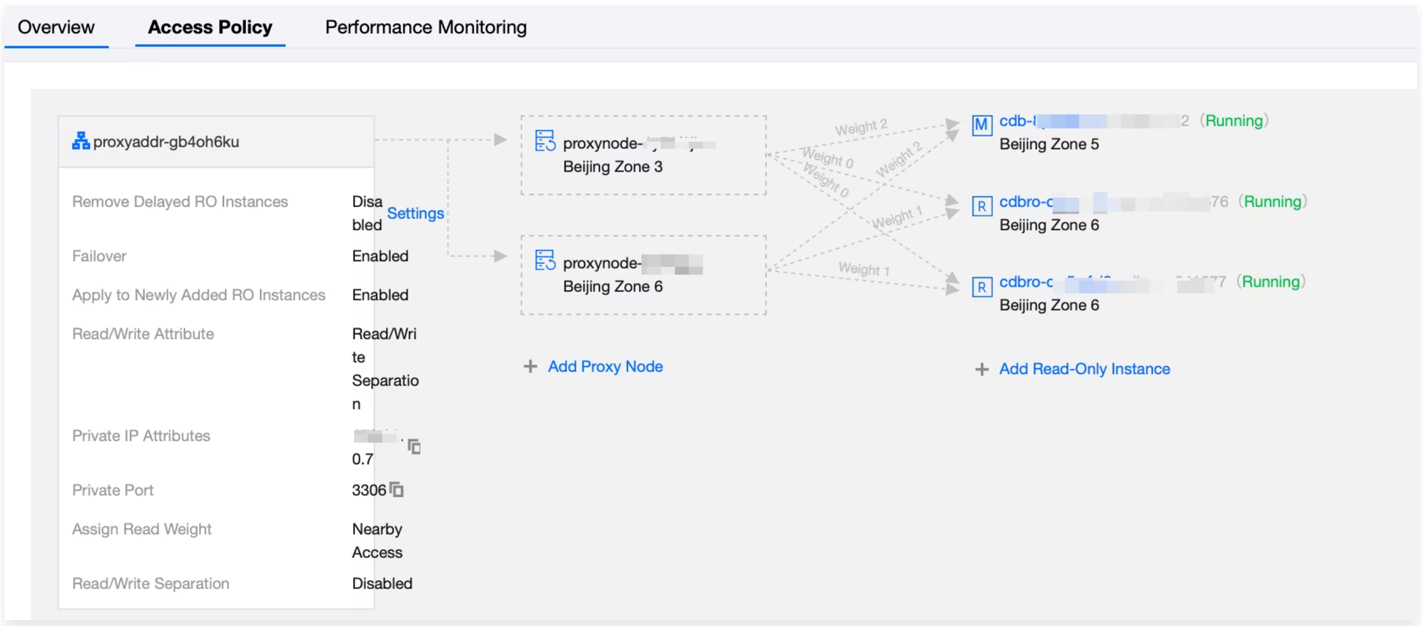
- Only the running read-write instance and read-only instances can be added to the database proxy.
- Currently, remote or delayed read-only instances cannot be mounted to the database proxy.

Category	Note
Read-Write Attribute	Select **Read/Write Separation**
Remove Delayed RO Instances	Set the **Remove Delayed RO Instances** policy. After this option is enabled, you can set **Delay Threshold** and **Least RO Instances**. The system will try removing or restoring a failed read-only instance no matter whether this option is enabled. • **Delay Threshold**: Enter an integer greater than or equal to 1 (in seconds). • **Least RO Instances**: It is subject to the number of read-only instances owned by the source instance. If it is set to `0`, when all read-only nodes are removed, all read requests will be routed to the source instance until at least one of the removed read-only instances rejoins the database proxy to continue processing the read requests.
Connection Pool Status	The connection pool feature mainly mitigates the instance load caused by frequent new connections in non-persistent connection business scenarios. After this option is enabled, you can select the supported connection pool type, which currently can only be the session-level connection pool by default.
Transaction Split	You can set whether to enable this feature. After it is enabled, reads and writes in one transaction will be separated to different instances for execution, and read requests will be forwarded to read-only instances to reduce the load of the source instance.
Assign Read Weight	You can select **Assigned by system** or **Custom**. If multiple AZs are configured when the database proxy is enabled, you can separately assign the weights of proxy nodes in different AZs.
Failover (with **Read-Write Attribute** being **Read/Write Separation**)	You can set whether to enable this feature. After it is enabled, if database proxy fails, the database proxy address will route requests to the source instance.
Apply to Newly Added RO Instances	Enable or disable this parameter. After it is enabled, if you purchase new read-only instances, they will be automatically added to the database proxy. • If **Assign Read Weight** is set to **Assigned by system**, newly purchased read-only instances will be assigned with the default weight based on their specification. • If **Assign Read Weight** is set to **Custom**, when newly

	purchased read-only instances are added to the RO group, their weights will be 0 by default, which can be modified in the configuration of the database proxy's connection address.

Database Proxy Read/Write Separation Enabled

After the database proxy read/write separation feature is successfully enabled, the database proxy page is as follows:



Remove Delayed RO Instances	Disabled
Failover	Enabled
Apply to Newly Added RO Instances	Enabled
Read/Write Attribute	Read/Write Separation
Private IP Attributes	0.7
Private Port	3306
Assign Read Weight	Nearby Access
Read/Write Separation	Disabled

Documentation

- Configure Database Proxy Connection Address
- Switching Database Proxy Network
- Enable and disable connection pool feature

Building LAMP Stack for Web Application

Last updated：2023-09-01 17:37:45

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 highly available 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 the same database from multiple servers for simplified database operation and maintenance, so that you no longer need to worry about database installation, deployment, version update, and troubleshooting, etc.

Note
The TencentDB and CVM instances used in the tutorial reside in the same region. If this is not the case, see Connecting to MySQL Instance.

Initializing the TencentDB Instance

For more information on how to purchase and initialize TencentDB instances, see Purchase Methods and Creating MySQL Instance.

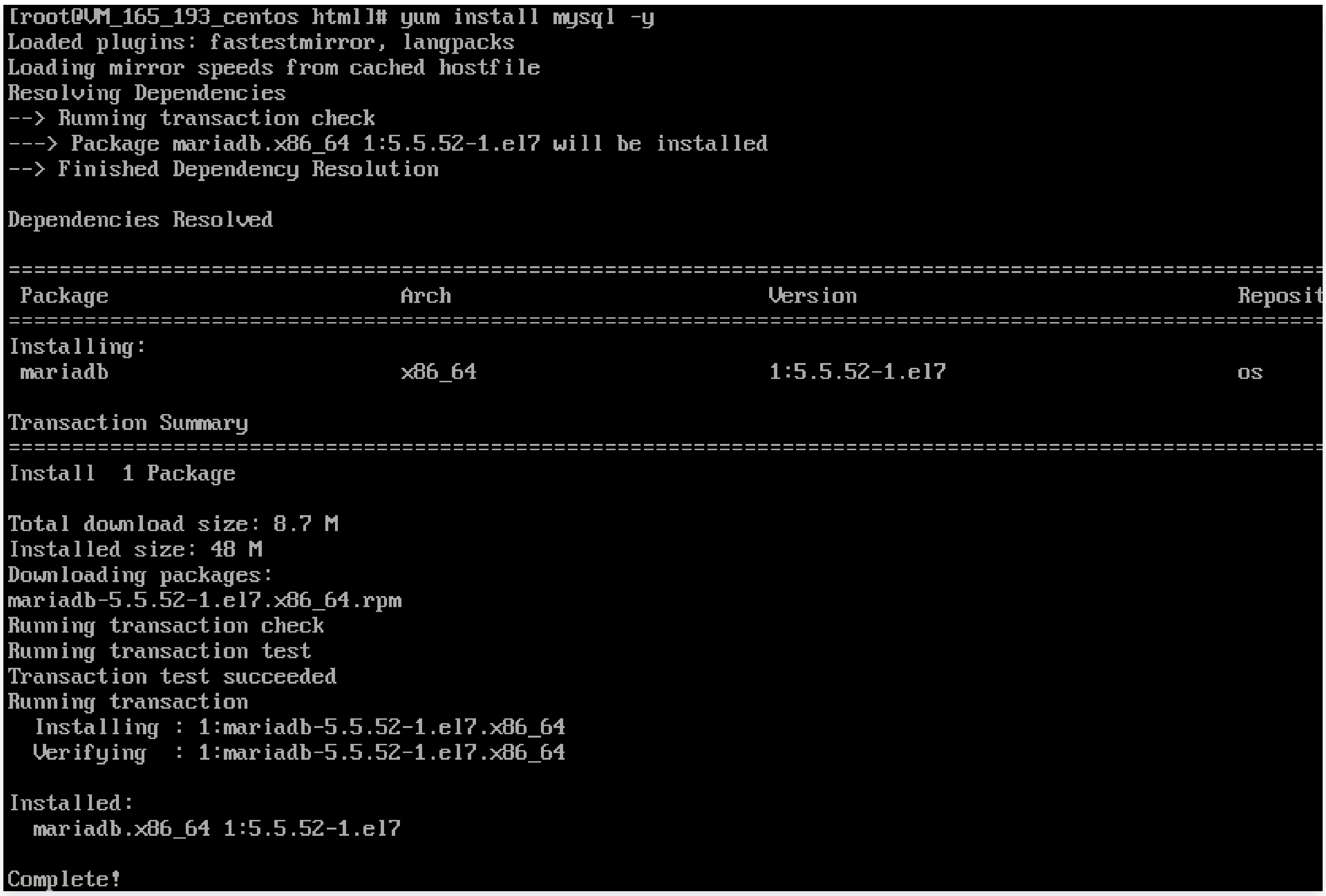
Logging in to the CVM Instance

For more information on how to purchase and access CVM instances, see Customizing Linux CVM Configurations. CentOS is used in this tutorial.

Installing the MySQL Client

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

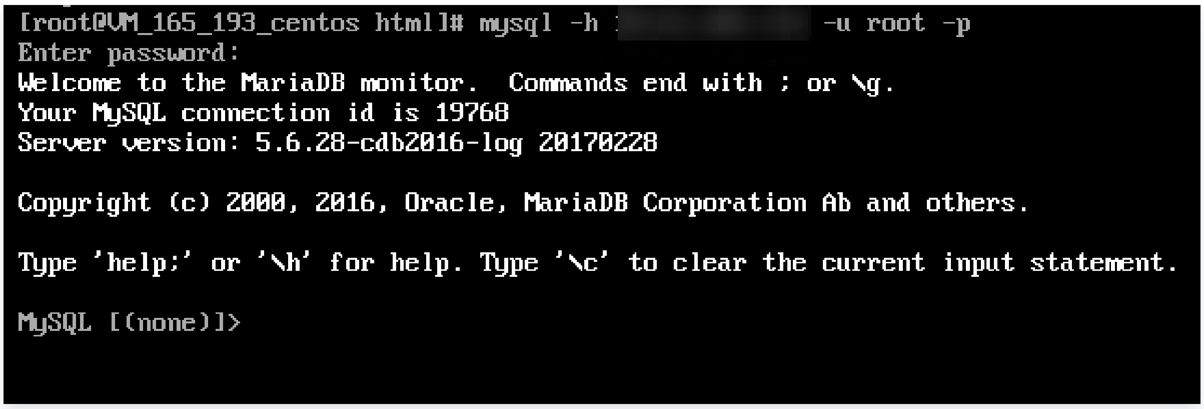
```
yum install mysql -y
```



```
[root@VM_165_193_centos html]# yum install mysql -y
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
---> Package mariadb.x86_64 1:5.5.52-1.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

==========================================================================================
 Package                       Arch                         Version                           Reposit
==========================================================================================
Installing:
 mariadb                       x86_64                       1:5.5.52-1.el7                    os

Transaction Summary
==========================================================================================
Install  1 Package

Total download size: 8.7 M
Installed size: 48 M
Downloading packages:
mariadb-5.5.52-1.el7.x86_64.rpm
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : 1:mariadb-5.5.52-1.el7.x86_64
  Verifying  : 1:mariadb-5.5.52-1.el7.x86_64

Installed:
  mariadb.x86_64 1:5.5.52-1.el7

Complete!
```

2. Connect to the TencentDB instance after the installation is completed.

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



```
[root@VM_165_193_centos html]# mysql -h [redacted] -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 19768
Server version: 5.6.28-cdb2016-log 20170228

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

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

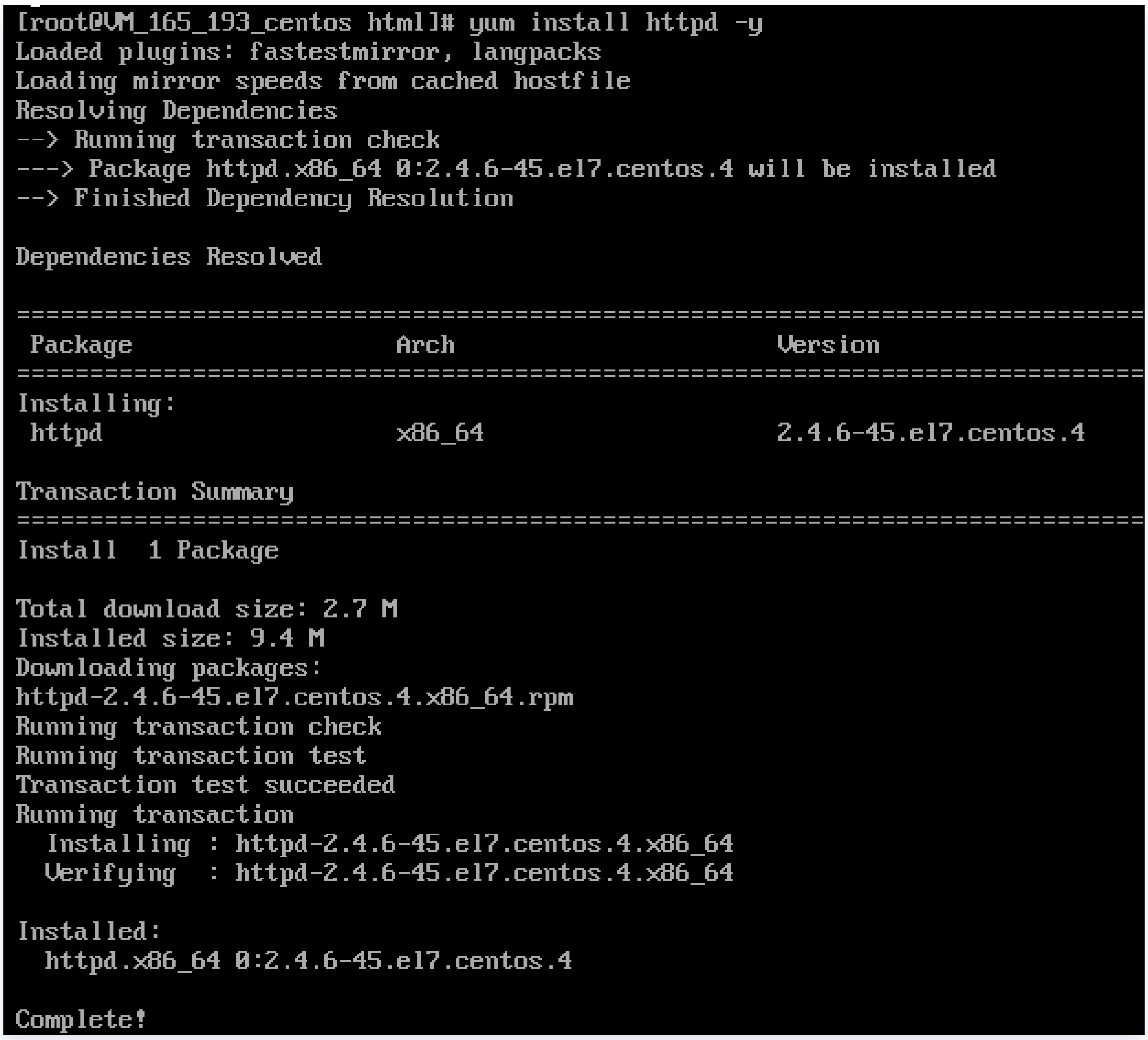
3. After the connection is successful, you can close the instance and proceed to the next step.

```
quit;
```

Installing the Apache service

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

```
yum install httpd -y
```



```
[root@VM_165_193_centos html]# yum install httpd -y
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.4.6-45.el7.centos.4 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=============================================================================
 Package                    Arch                       Version
=============================================================================
Installing:
 httpd                      x86_64                     2.4.6-45.el7.centos.4

Transaction Summary
=============================================================================
Install  1 Package

Total download size: 2.7 M
Installed size: 9.4 M
Downloading packages:
httpd-2.4.6-45.el7.centos.4.x86_64.rpm
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : httpd-2.4.6-45.el7.centos.4.x86_64
  Verifying  : httpd-2.4.6-45.el7.centos.4.x86_64

Installed:
  httpd.x86_64 0:2.4.6-45.el7.centos.4

Complete!
```

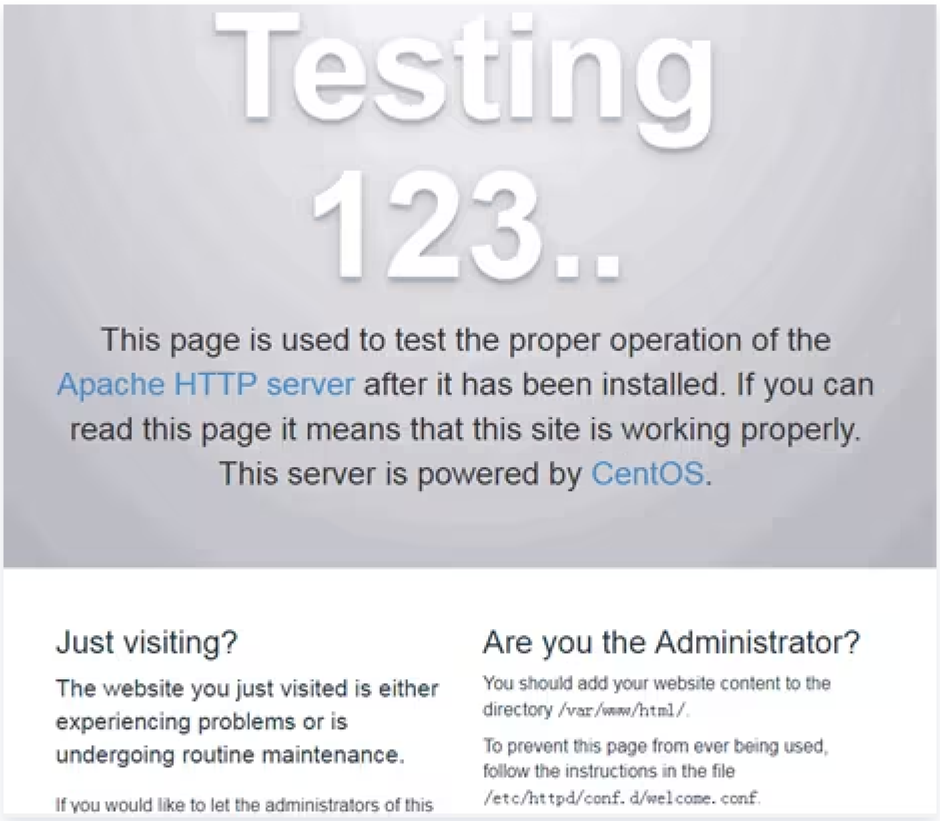
2. Launch the Apache service.

```
service httpd start
```

3. Test Apache.

> **Note**
> 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://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 started successfully.



Testing 123..

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?

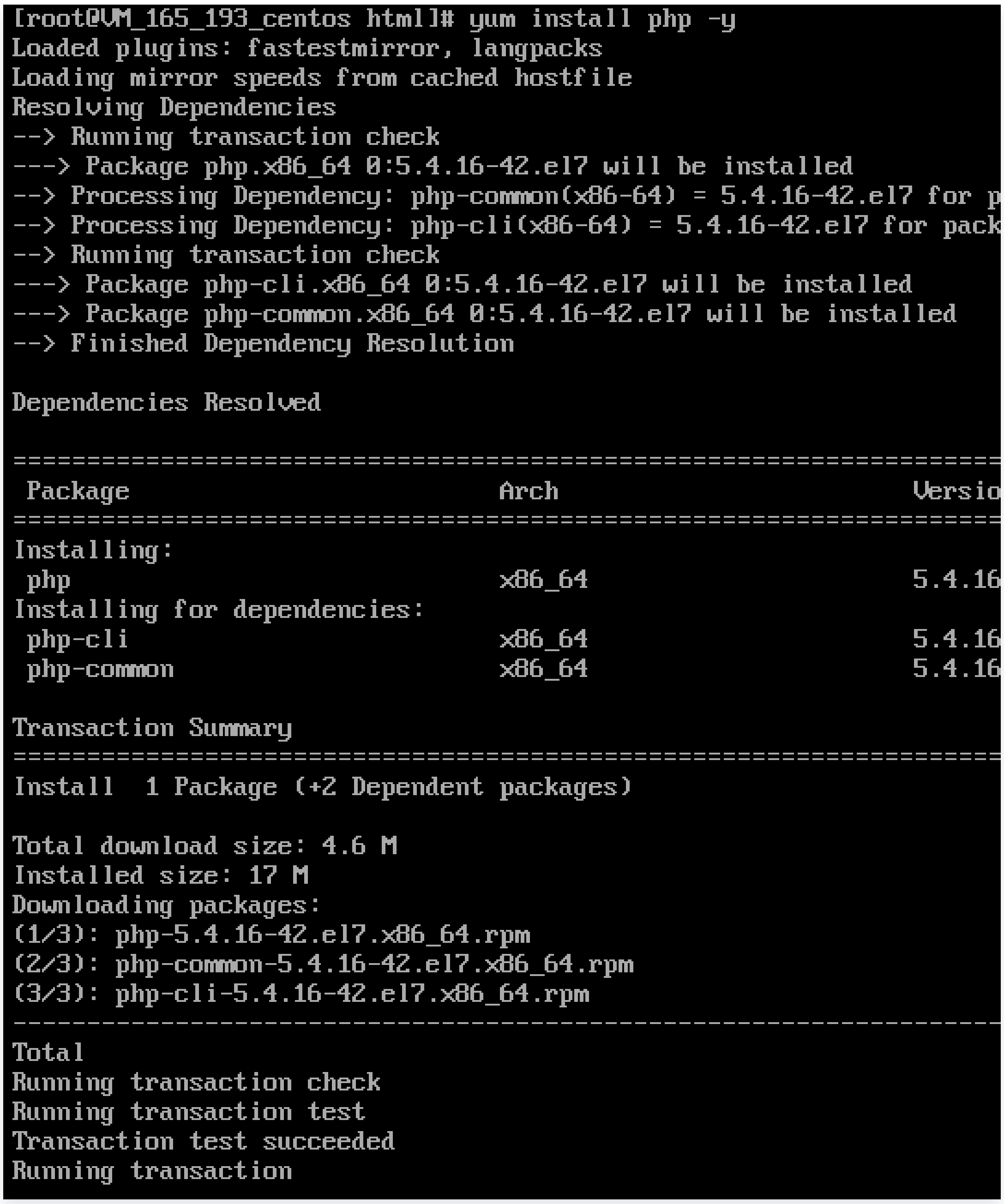
You should add your website content to the directory /var/www/html/.

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 in the CVM instance with `yum`.

```
yum install php -y
```



```
[root@VM_165_193_centos html]# yum install php -y
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
---> Package php.x86_64 0:5.4.16-42.el7 will be installed
--> Processing Dependency: php-common(x86-64) = 5.4.16-42.el7 for p
--> Processing Dependency: php-cli(x86-64) = 5.4.16-42.el7 for pack
--> Running transaction check
---> Package php-cli.x86_64 0:5.4.16-42.el7 will be installed
---> Package php-common.x86_64 0:5.4.16-42.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

===================================================================
 Package                          Arch                         Versio
===================================================================
Installing:
 php                              x86_64                       5.4.16
Installing for dependencies:
 php-cli                          x86_64                       5.4.16
 php-common                       x86_64                       5.4.16

Transaction Summary
===================================================================
Install  1 Package (+2 Dependent packages)

Total download size: 4.6 M
Installed size: 17 M
Downloading packages:
(1/3): php-5.4.16-42.el7.x86_64.rpm
(2/3): php-common-5.4.16-42.el7.x86_64.rpm
(3/3): php-cli-5.4.16-42.el7.x86_64.rpm
-------------------------------------------------------------------
Total
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
```

Creating a Project to Test the LAMP Environment

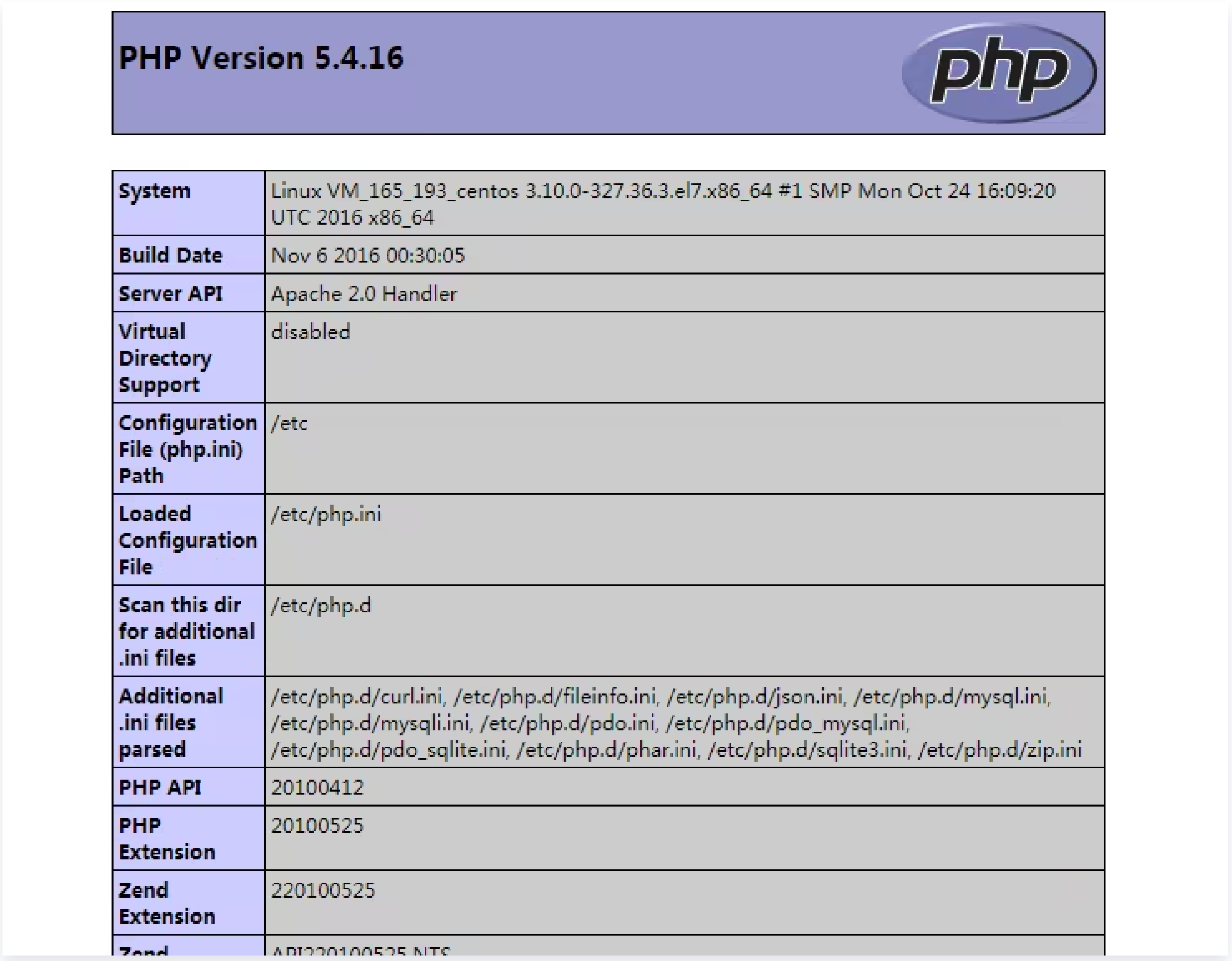
1. Create an info.php file in the `/var/www/html` directory of the CVM instance. Below is the sample code:

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

2. Restart the Apache service.

```
service httpd restart
```

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



PHP Version 5.4.16

System	Linux VM_165_193_centos 3.10.0-327.36.3.el7.x86_64 #1 SMP Mon Oct 24 16:09:20 UTC 2016 x86_64
Build Date	Nov 6 2016 00:30:05
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/curl.ini, /etc/php.d/fileinfo.ini, /etc/php.d/json.ini, /etc/php.d/mysql.ini, /etc/php.d/mysqli.ini, /etc/php.d/pdo.ini, /etc/php.d/pdo_mysql.ini, /etc/php.d/pdo_sqlite.ini, /etc/php.d/phar.ini, /etc/php.d/sqlite3.ini, /etc/php.d/zip.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend	API220100525,NTS

Utilizing MySQL API with Python Language
Instance Purchase

Last updated：2023-09-01 17:39:13

API	Description
CreateDBInstance	Creates monthly-subscribed TencentDB for MongoDB instances
CreateDBInstanceHour	Create a pay-as-you-go TencentDB instance
DescribeDBInstances	Querying instance list
DescribeDBPrice	Queries the prices of TencentDB instances

CreateDBInstance for Creating a Monthly Subscribed TencentDB Instance

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

# Introduce the Cloud 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


'''Purchase a master instance'''
def CreateDBInstancedemomaster():
    try:
        Instantiate an authentication object, passing in the Tencent
Cloud account SecretId and SecretKey as parameters
        cred = credential.Credential("secretId", "secretKey")
        Instantiate a client object for the desired product (using cdb
as an example)
        client = cdb_client.CdbClient(cred, "ap-beijing")

        Instantiate a request object: req =
models.ModifyInstanceParamRequest()
        req = models.CreateDBInstanceRequest()
        req.Memory =2000
        req.Volume = 120
        req.Period=1
        req.GoodsNum = 1
        req.Zone = "ap-beijing-1"
        req.Port = 3306
        #req.MasterInstanceId = "cdb-7ghaiocc"
        req.InstanceRole = "master"
        req.EngineVersion = "5.6"
        req.Password = "CDB@Qcloud"
        req.ProtectMode = 0
        req.InstanceName = "tencentcdb"
        req.SecurityGroup = ["sg-eq0hvlzp"]


        Invoke the desired API through the client object by passing in
the request object
        resp = client.CreateDBInstance(req)
        Returns a JSON-formatted response string
        print(resp.to_json_string())


    except TencentCloudSDKException as err:
        msg = traceback.format_exc() # Method 1
        print (msg)




'''Purchase a read-only instance'''
def CreateDBInstancedemoro():
    try:
        Instantiate an authentication object, passing in the Tencent
Cloud account SecretId and SecretKey as parameters
        cred = credential.Credential("secretId", "secretId")

        Instantiate a client object for the desired product (using cdb
as an example)
        client = cdb_client.CdbClient(cred, "ap-beijing")

        Instantiate a request object: req =
models.ModifyInstanceParamRequest()
        req = models.CreateDBInstanceRequest()
        req.Memory =2000
        req.Volume = 200
        req.Period=1
        req.GoodsNum = 1
        req.Zone = "ap-beijing-1"
        req.Port = 3306
        req.InstanceRole = "ro"
        req.EngineVersion = "5.6"
        req.Password = "CDB@Qcloud"
        req.ProtectMode = 0
        req.DeployMode =1
        req.GoodsNum =2
        req.SlaveZone = "ap-beijing-1"
        req.ParamList = [{"name":"max_connections","value":"1000"},
{"name":"lower_case_table_names","value":"1"}]
        req.BackupZone = "0"
        req.AutoRenewFlag = 0
        req.MasterInstanceId ="cdb-bgr97hu0"
        req.RoGroup = {"RoGroupMode":"allinone","RoGroupName":"roweek"}
        req.InstanceName = "tencentcdbRO"


        Invoke the desired API through the client object by passing in
the request object
        resp = client.CreateDBInstance(req)
        Returns a JSON-formatted response string
        print(resp.to_json_string())
```

```
    except TencentCloudSDKException as err:
        msg = traceback.format_exc() # Method 1
        print (msg)


'''Purchase a disaster recovery instance'''
def CreateDBInstancedemodr():
    try:
        Instantiate an authentication object, passing in the Tencent
Cloud account SecretId and SecretKey as parameters
        cred = credential.Credential("secretId", "secretKey")

        Instantiate a client object for the desired product (using cdb
as an example)
        client = cdb_client.CdbClient(cred, "ap-shanghai")

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

        req.Memory = 4000
        req.Volume = 200
        req.Period=1
        req.GoodsNum = 1
        #req.Zone = "ap-shanghai-2"
        req.Port = 3306
        req.InstanceRole = "dr"
        #req.MasterInstanceId
        req.EngineVersion = "5.6"
        req.Password = "CDB@Qcloud"
        req.ProtectMode = 0
        req.DeployMode =0
        #req.SlaveZone = "ap-guangzhou-3"
        req.ParamList = [{"name":"max_connections","value":"1000"},
{"name":"lower_case_table_names","value":"1"}]
        req.BackupZone = "0"
        req.AutoRenewFlag = 0
        #req.RoGroup = {"RoGroupMode":"alone","RoGroupName":"roweek"}
        #req.RoGroup = {"RoGroupName":"roweek"}
        #param = models.RoGroup()
        #param.RoGroupMode = "alone"
        #param.RoGroupName = "roweek"
        #param.MinRoInGroup = 1
        #req.RoGroup = [param]


        #ro = [{"roGroupMode":"allinone"},{"RoGroupName":"ro_www"}]
        #req.RoGroup = [ro]
        req.MasterInstanceId ="cdb-bgr97hu0"
        req.MasterRegion = "ap-beijing"
        #roGroup = [RoGroupMode="allinone",
RoGroupName="weekro",RoOfflineDelay=1,MinRoInGroup=5,MinRoInGroup=1]
        #req.RoGroup = [roGroup]
        req.InstanceName = "tencentcdbDR"


        Invoke the desired API through the client object by passing in
the request object
        resp = client.CreateDBInstance(req)
        Returns a JSON-formatted response string
        print(resp.to_json_string())


    except TencentCloudSDKException as err:
        msg = traceback.format_exc() # Method 1
        print (msg)

#CreateDBInstancedemodr()
#CreateDBInstancedemoro()
#CreateDBInstancedemomaster()
```

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 -*-
# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb 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
    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


    Invoke the desired API through the client object by passing in the
request object
```

```
    resp = client.CreateDBInstanceHour(req)

    Returns a JSON-formatted response string
    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 -*-

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb 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"]
    Invoke the desired API through the client object by passing in the
request object
    resp = client.DescribeDBInstances(req)

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
        msg = traceback.format_exc() # Method 1
        print (msg)
```

DescribeDBPrice: Query the prices of TencentDB instances

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

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb 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


    Invoke the desired API through the client object by passing in the
request object
    resp = client.DescribeDBPrice(req)

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

Instance management

Last updated：2023-09-01 17:39:52

API	Description
ModifyInstanceParam	Modifies instance parameters.
CloseWanService	Disabling public network access for an instance
OpenWanService	Enabling public network access for an instance
RestartDBInstances	Restart an instance
OpenDBInstanceGTID	Enables GTID for an instance
ModifyDBInstanceName	Renaming a TencentDB instance
ModifyDBInstanceProject	Modifying the project to which a TencentDB instance belongs
ModifyDBInstanceVipVport	Modifies the IP and port number of a TencentDB instance
DescribeDBInstanceCharset	Querying the character set of a TencentDB instance
DescribeDBInstanceConfig	Querying the configuration information of a TencentDB instance
DescribeDBInstanceGTID	Queries whether GTID is enabled for a TencentDB instance
DescribeDBInstanceRebootTime	Querying the estimated restart time of a TencentDB instance

ModifyInstanceParam for Modifying Instance Parameters

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


# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb as an
example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    # Instantiate a request object
    req = models.ModifyInstanceParamRequest()
    req.InstanceIds = ["cdb-1y6g3zj8","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"
    #param.CurrentValue = "1000"
    #req.ParamList = [param]



    print req
    Invoke the desired API through the client object by passing in the
request object
    resp = client.ModifyInstanceParam(req)

    Returns a JSON-formatted response string
    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 -*-

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb as an
example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

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

    Invoke the desired API through the client object by passing in the
request object
    resp = client.CloseWanService(req)

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

OpenWanService for Enabling Public Network Access for an Instance

```
#!/usr/bin/python
```

```
# -*- coding: utf-8 -*-

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb as an
example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

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

    Invoke the desired API through the client object by passing in the
request object
    resp = client.OpenWanService(req)

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

RestartDBInstances for Restarting Instances

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

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb as an
example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

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

    Invoke the desired API through the client object by passing in the
request object
    resp = client.RestartDBInstances(req)

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

OpenDBInstanceGTID for Enabling GTID for an Instance

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

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb as an
example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

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



    Invoke the desired API through the client object by passing in the
request object
    resp = client.OpenDBInstanceGTID(req)

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

ModifyDBInstanceName for Renaming a TencentDB Instance

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

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb 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 = "1sChinese"

    Invoke the desired API through the client object by passing in the
request object
    resp = client.ModifyDBInstanceName(req)

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

ModifyDBInstanceProject for Modifying the Project to Which a TencentDB Instance Belongs

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

# Introduce the Cloud 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, passing in the Tencent
Cloud account SecretId and SecretKey as parameters
        cred = credential.Credential("secretId", "secretKey")

        Instantiate a client object for the desired product (using cdb
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



        Invoke the desired API through the client object by passing in
the request object
        resp = client.ModifyDBInstanceProject(req)

        Returns a JSON-formatted response string
        print(resp.to_json_string())
    except TencentCloudSDKException as err:
        msg = traceback.format_exc() # Method 1
        print (msg)




DescribeDBInstancesList()
```

ModifyDBInstanceVipVport for Modifying the IP and Port Number of a TencentDB Instance

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb 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

    Invoke the desired API through the client object by passing in the
request object
    resp = client.ModifyDBInstanceVipVport(req)

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
        msg = traceback.format_exc() # Method 1
        print (msg)
```

DescribeDBInstanceCharset for Querying the Character Set of a TencentDB Instance

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

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb as an
example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

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


    Invoke the desired API through the client object by passing in the
request object
    resp = client.DescribeDBInstanceCharset(req)

    Returns a JSON-formatted response string
    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 -*-

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb as an
example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

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


    Invoke the desired API through the client object by passing in the
request object
    resp = client.DescribeDBInstanceConfig(req)

    Returns a JSON-formatted response string
    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 -*-

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb as an
example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

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


    Invoke the desired API through the client object by passing in the
request object
    resp = client.DescribeDBInstanceGTID(req)

    Returns a JSON-formatted response string
    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 -*-

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb as an
example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    Instantiate a request object: req =
models.ModifyInstanceParamRequest()
    req = models.DescribeDBInstanceRebootTimeRequest()
    req.InstanceIds = ["cdb-1y6g3zj8"]


    Invoke the desired API through the client object by passing in the
request object
    resp = client.DescribeDBInstanceRebootTime(req)

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

Backup Task

Last updated: 2023-09-01 17:40:33

API	Description
CreateBackup	Creates a TencentDB instance backup
DeleteBackup	Deleting a TencentDB instance backup
DescribeBackupConfig	This API shows you how to query the configuration information of a TencentDB instance backup.
DescribeBackups	This example shows you how to query the list of data backup files.
DescribeBinlogs	This example shows you how to query a binlog.
DescribeSlowLogs	Querying slow logs
ModifyBackupConfig	This example shows you how to modify the database backup configuration.

CreateBackup: Creating a TencentDB instance backup

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

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")
    Instantiate a client object for the desired product (using cdb 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
    Invoke the desired API through the client object by passing in the
request object
    resp = client.CreateBackup(req)

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

DeleteBackup: Deleting a TencentDB instance backup

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

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb 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


    Invoke the desired API through the client object by passing in the
request object
    resp = client.DeleteBackup(req)

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

DescribeBackupConfig: Querying the configuration information of a TencentDB instance backup

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

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")
    Instantiate a client object for the desired product (using cdb 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
    Invoke the desired API through the client object by passing in the
request object
    resp = client.DescribeBackupConfig(req)

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

DescribeBackups: Query the list of data backup files

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

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb as an
example)
```

```
    client = cdb_client.CdbClient(cred, "ap-shanghai")

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



    Invoke the desired API through the client object by passing in the
request object
    resp = client.DescribeBackups(req)
    print resp

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

DescribeBinlogs: Querying binary logs

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

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb as an
example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")
    Instantiate a request object: req =
models.ModifyInstanceParamRequest()
    req = models.DescribeBinlogsRequest()
    req.InstanceId = "cdb-7ghaiocc"


    Invoke the desired API through the client object by passing in the
request object
    resp = client.DescribeBinlogs(req)

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

DescribeSlowLogs: Querying slow query logs

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

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb as an
example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")
    Instantiate a request object: req =
models.ModifyInstanceParamRequest()
    req = models.DescribeSlowLogsRequest()
    req.InstanceId = "cdb-7ghaiocc"


    Invoke the desired API through the client object by passing in the
request object
    resp = client.DescribeSlowLogs(req)

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

ModifyBackupConfig: Modify the TencentDB instance backup configuration

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

# Introduce the Cloud 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, passing in the Tencent Cloud
account SecretId and SecretKey as parameters
    cred = credential.Credential("secretId", "secretKey")

    Instantiate a client object for the desired product (using cdb as an
example)
    client = cdb_client.CdbClient(cred, "ap-shanghai")
    Instantiate a request object: req =
models.ModifyInstanceParamRequest()
    req = models.ModifyBackupConfigRequest()
    req.InstanceId = "cdb-1y6g3zj8"
    req.ExpireDays = 10
    req.StartTime = "06:00-10:00"
    req.BackupMethod = "logical"
    print req


    Invoke the desired API through the client object by passing in the
request object
    resp = client.ModifyBackupConfig(req)

    Returns a JSON-formatted response string
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

The primary and secondary instances have inconsistent query data

Last updated：2026-03-10 19:11:48

This document introduces the possible causes and solutions for adding an auto-increment primary key that causes inconsistent query data in primary and secondary instances.

Issue Description

Querying with the same auto-increment primary key value (auto-increment ID) on both the primary and secondary instances results in inconsistent data in the query results.

Possible Causes

When an auto-increment primary key is added to a table without a primary key, the auto-increment primary key values are assigned based on the physical storage order of the data in the table. Because a table without a primary key lacks an explicit primary key constraint, the order of its row data is determined by the RowID within the storage engine, and the RowID for the same data may differ between the primary and secondary instances, resulting in inconsistent physical arrangement order of the data in the primary and secondary instances, thus causing the auto-increment primary key values assigned to the same data to differ across different instances. Therefore, when the primary and secondary instances are queried separately using the same auto-increment primary key value, the results obtained may differ. For details, see BUG#92949 and MySQL official documentation.

Solution

1. Create a new table identical to the original table without a primary key on the primary instance, and add an auto-increment primary key.
2. Sort the data by all fields and then insert it into the new table.
3. Delete the original table without a primary key, and rename the new table to the name of the original table without a primary key.

Example

```
CREATE TABLE t2 LIKE t1;
ALTER TABLE t2 ADD id INT AUTO_INCREMENT PRIMARY KEY;
INSERT INTO t2 SELECT * FROM t1 ORDER BY col1, col2;
```

```
DROP TABLE t1;
RENAME TABLE t2 TO t1;
```