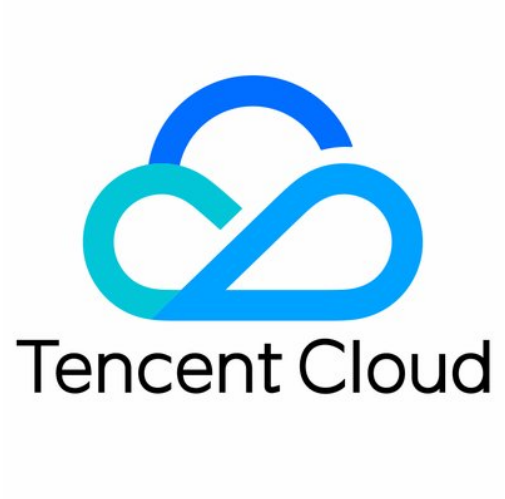
TencentDB for MySQL

プラクティスチュートリアル

製品ドキュメント



カタログ：

プラクティスチュートリアル
MySQL利用規約

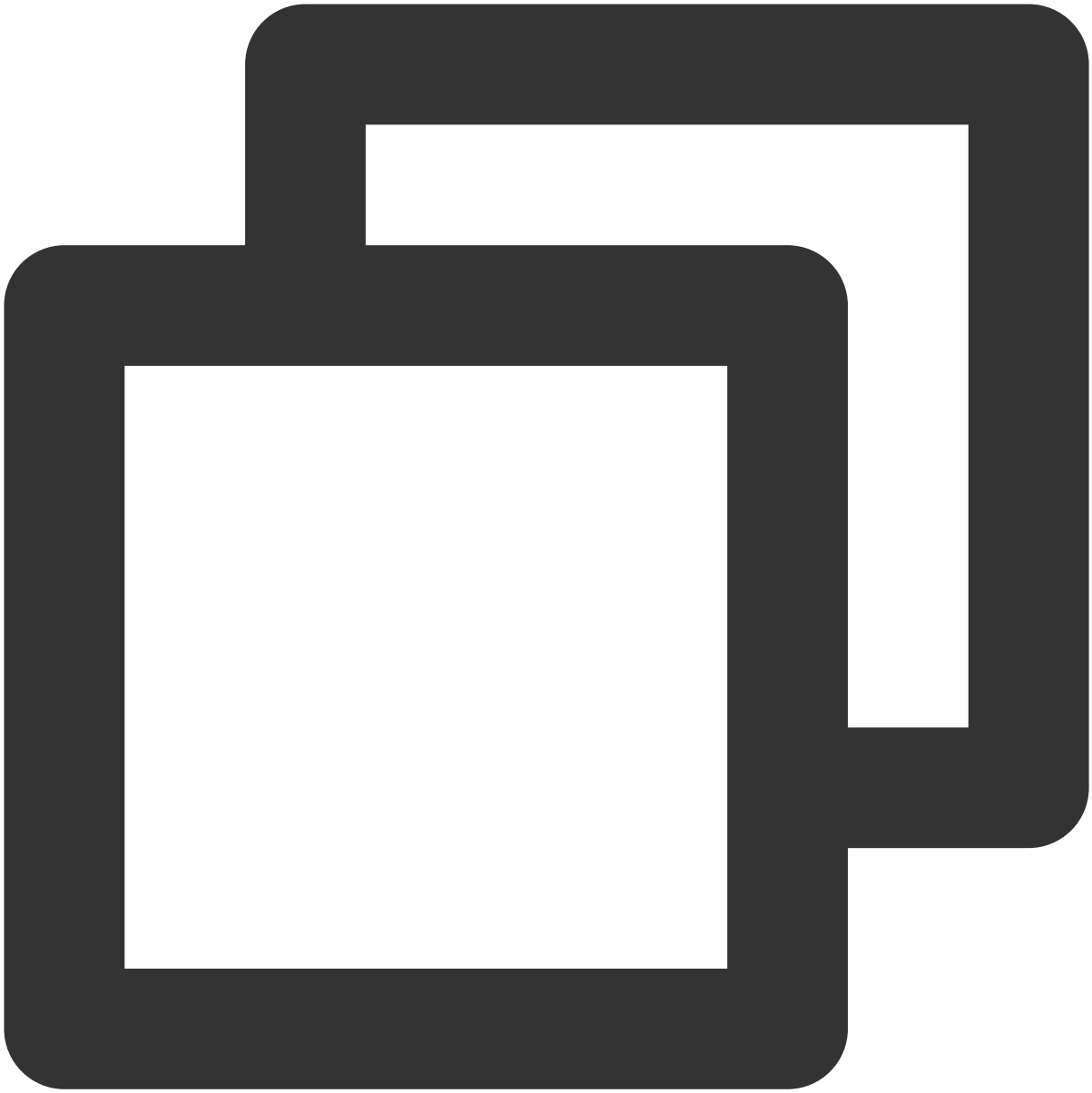
最終更新日：：2024-07-25 17:56:18

目的

不適切な操作によってTencentDB for MySQLが使用できなくなることを防ぐため、TencentDB for MySQLの管理とメンテナンスを標準化します。
SQLを合理的に記述して、TencentDB for MySQLの最適なパフォーマンスを引き出すよう、データベース開発者に指示します。

権限管理基準

TencentDB for MySQLの安定性とセキュリティを考慮し、TencentDB for MySQLはsuper、shutdown、file権限を制限しています。TencentDB for MySQLでsetセンテンスを実行すると、次のエラーが表示される場合があります。



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

ソリューション：関連パラメータの変更をsetする必要がある場合は、コンソールのインスタンス管理ページの**データベース管理>パラメータ設定**機能を使用して完了することができます。
必要に応じて権限を付与し、一般的なアプリケーションプログラムはDML（SELECT、UPDATE、INSERT、DELETE）権限のみを付与します。
権限付与の対象を最小限にするという原則により、一般的なアプリケーションプログラムアクセスユーザーにはデータベースレベルに応じて権限を付与します。

権限を付与されたユーザーがアクセスする場合は、特定のIPまたはIPセグメントへのアクセスのみが許可され、コンソールでセキュリティグループを設定することによってアクセスを制限することができます。セキュリティグループの設定は、コンソールが提示する基準に従って操作する必要があります。パブリックネットワークアクセスでセキュリティグループを設定する場合は、関連するすべてのegress IP を必ずオープンにしてください。
管理アカウントと開発アカウントを分離します。

日常的な操作の基準

注意事項

データベースインスタンスのセキュリティを向上させるため、脆弱なパスワードの使用を禁止します。
イントラネット接続ログインでは、client側のクラウドサーバCVMとクラウドデータベースMySQLが同じアカウントの同じ地域および同じVPCのマシンであることを確認する必要があります。
コンソールからダウンロードしたbinlogログをローカルで解析する必要がある場合は、クライアントのMySQLバージョンがTencentDB for MySQLのインスタンスバージョンと一致していることを確認する必要があります。一致していない場合は、解析時に文字化けが発生する恐れがあります。バージョン3.4以降のmysqlbinlogを使用することをお勧めします。
コンソールでプライベートネットワークを介してCVMのコールドバックアップファイルをダウンロードする場合は、urlを引用符で囲んでください。この操作を行わないと、404エラーが表示されます。

推奨事項

業務ピーク時にonline ddl操作を極力行わないようにしてください。利用可能なツールについては、`pt-online-schema-change` をご参照ください。
業務ピーク時にデータのバッチ操作を行うことは極力避けてください。業務オフピーク時がバッチ操作に最適です。
単独のインスタンスで複数の業務を実行することを避けてください。カップリングレベルが高すぎると、業務間で相互に影響を及ぼし合うリスクがあります。
誤操作によるデータ紛失のリスクを低減するため、トランザクションの自動送信をオフにし、オンライン操作での `begin;` を習慣付けることをお勧めします。誤操作については、TencentDB for MySQLのロールバック機能を使用することもできます (現在、5日以内の任意の時点のロールバックをサポートしています)。関連のテーブルにクロスデータベースやクロステーブルのロジックが含まれない場合は、高速ロールバックまたは超高速ロールバックを使用してデータをより迅速に回復することができます。ロールバックによって新しく生成されたデータベーステーブル名のデフォルトは `データベーステーブル名_bak` です。
プロモーション活動を行う業務などについては、事前にリソースを見積もり、インスタンスの最適化を行ってください。需要量が大きい場合は、適宜、対応するサービスマネージャーにご連絡ください。

データベーステーブル設計仕様

注意事項

TencentDB for MySQL 5.6以降のバージョンは、MyISAMおよびMemoryエンジンをサポートしていません。Memoryエンジンのニーズがある場合は、TencentDB for RedisおよびMemcachedを使用することをお勧めします。
自作データベースをTencentDB for MySQLに移行すると、MyISAMエンジンはInnoDBエンジンに自動的に変換されます。
自動インクリメント列のあるテーブルについては、自動インクリメント列に少なくとも1つの個別のインデックス、または自動インクリメント列で始まる1つの複合インデックスが必要です。
`row_format` は非fixedを保証する必要があります。
各テーブルにはプライマリキーが必要です。適切な列をプライマリキーとして選択できなくとも、無意味な列をプライマリキーとして追加する必要があります。MySQL第1パラダイム標準InnoDBセカンダリインデックスのリーフノードは、プライマリキー値を格納することができます。インデックスが占有するディスク領域を削減し、効率を向上させるため、自動インクリメント列をプライマリキーとして使用することをお勧めします。 `binlog_format` がrowに設定されている場合、プライマリキーのないデータの一括削除は深刻なマスター/スレーブ遅延を引き起こす恐れがあります。
フィールドは極力NOT NULLと定義し、デフォルト値を追加します。NULLはSQL開発に多くの問題をもたらし、インデックスを作成できなくなります。NULLを計算する場合は、IS NULLとIS NOT NULLのみを判断に使用することができます。

推奨事項

業務シナリオ分析とデータアクセス (データベースの読み書きQPS、TPS、Storageキャパシティなどを含みます) の見積もりを通じて、データベースの使用リソースを合理的に計画し、コンソールBasic Cloud Monitorインターフェースで、TencentDB for MySQLインスタンスの各種監視を設定することもできます。
データベース構築の原則は、同種の業務のテーブルを1つのデータベースに格納し、異なる業務のテーブルに同じデータベースを共用することをできる限り避け、プログラムによるデータベース間の関連付け操作を極力実行しないようにすることです。この操作は後続の高速ロールバックにも一定の影響を与える恐れがあります。
文字化けのリスクを低減するため、文字セットにはutf8mb4を統一して使用します。一部の複雑な漢字とemojiの表情を正しく表示するためにはutf8mb4を使用する必要があります。文字セットの変更は、変更後に作成されたテーブルでのみ有効です。したがってTencentDB for MySQLを新規購入し、インスタンスを初期化する際にutf8mb4を選択することをお勧めします。
少数のフィールドについては、decimal型の使用をお勧めします。floatやdoubleは精度が充分ではなく、特に金銭に関わる業務ではdecimalを使用する必要があります。
サイズの大きなテキスト、バイナリーデータ、画像、ファイルなどをtext/blob を使用してデータベースに保存することを極力避けてください。これらのデータはローカルディスクファイルとして保存し、データベースにはインデックス情報のみを保存してください。
外部キーを極力使用しないようにしてください。アプリケーション層に外部キーのロジックを実装することをお勧めします。外部キーとカスケード更新は同時実行性の高いシナリオには不向きです。インサートのパフォーマンスが低下し、大量に同時実行する場合にデッドロックが発生します。

業務ロジックとデータストレージ間のカップリングレベルを低下させます。データベースは主にデータを格納し、業務ロジックは可能な限りアプリケーション層を介して実装されます。ストアドプロシージャ、トリガー、関数、event、ビューなどの高度な機能の使用を最小限に抑えます。これらの機能は移植性、拡張性に劣ることから、インスタンスにこれらのオブジェクトが存在する場合は、移行アカウントとdefinerの不一致による移行の失敗を避けるため、デフォルトでdefinerを設定しないことをお勧めします。
短期的な業務は大きな規模に至らないことから、パーティションテーブルの使用を禁止することをお勧めします。パーティションテーブルは、主にアーカイブ管理に使用され、宅配便業界やeコマース業界のオーダーシートとして多く使用されます。業務における80%以上のクエリーがパーティションフィールドを使用しない限り、パーティションテーブルはパフォーマンスを向上させません。
読み取り負荷が高く、かつ整合性要件が低い(秒レベルの遅延でデータを受信する) 業務シナリオでは、ライブラリから読み取り専用インスタンスを購入して、読み取り/書き込み分離ポリシーを実装することをお勧めします。

インデックス設計仕様

注意事項

頻繁に更新され、識別度の高くない列にインデックスを作成することは禁止されています。レコードの更新によってB+ツリーを変更することができます。頻繁に更新されるフィールドにインデックスを作成すると、データベースのパフォーマンスが大幅に低下します。
複合インデックスを作成する場合は、識別度が最も高い列をインデックスの左端に配置します。例えば、 `select xxx where a = x and b = x;` では、aとbを同時に使用してインデックスを作成し、aの識別度の方が高い場合は、 `idx_ab(a,b)` のように作成します。不等号と等号の混合判定条件がある場合は、等号条件の列を前方に配置します。例えば `where a xxx and b = xxx` では、aの識別度の方が高い場合でも、インデックスaを使用できないため、bをインデックスの最前列に配置する必要があります。

推奨事項

1つのテーブルのインデックス数が5を超えず、1つのインデックスのフィールド数が5を超えないことをお勧めします。数が多すぎるとフィルタリングの役割を果たせず、インデックスがスペースを占有し、管理するためのリソースも消費します。
業務においてSQLフィルタリングが最も多く、cardinality値が高い列を選択してインデックスを作成します。業務SQLが使用しない列にインデックスを作成しても無意味です。フィールドの一意性が高いほどcardinality値が高いことを意味し、インデックスのフィルタリング効果も高くなります。一般的なインデックス列のcardinalityレコード数は10%未満で、これが性別フィールドなどの効率の低いインデックスであると認識できます。
varcharフィールドにインデックスを作成する場合は、列全体で直接インデックスを作成するのではなく、インデックスの長さを指定することをお勧めします。通常、varchar列は長いため、特定の長さを指定してインデックスを作成することで、識別度を充分に高めることができます。列全体のインデックスを作成する必要はありません。列全体のインデックスを作成すると重くなり、インデックスのメンテナンスコストが増大します。

count(distinct left(列名, インデックスの長さ))/count(*)を使用して、インデックスの識別度を確認することができます。
冗長なインデックスを回避します。2つのインデックス(a, b) (a)が同時に存在する場合は、(a)が冗長なインデックスredundant indexに該当します。 クエリーのフィルタ条件がa列である場合は、(a, b)インデックスで足り、(a)インデックスを個別に作成する必要はありません。
カバリングインデックスを合理的に利用してIOオーバーヘッドを削減します。InnoDBではセカンダリインデックスのリーフノードは自身のキー値とプライマリキー値のみを格納します。SQLクエリーがインデックス列またはプライマリキーでない場合、このインデックスは最初に対応するプライマリキーを検出し、次にプライマリキーに従って、検出したい列を検出します。これがテーブルのリターンであり、追加のIOオーバーヘッドが発生します。この時、この問題を解決するために、カバリングインデックスを利用することができます。例えば、 `select a,b from xxx where a = xxx` で、aがプライマリキーでない場合、aとb2つの列の複合インデックスを作成すれば、テーブルのリターンが行われません。

SQL作成仕様

注意事項

UPDATE、DELETE操作にはLIMITを使用せず、WHEREを使用して正確に整合させる必要があります。LIMITがランダムである場合は、この操作によってデータエラーが発生する恐れがあります。
`INSERT INTO t_xxx VALUES(xxx)` の使用は禁止されており、テーブル構造の変更によるデータエラーを回避するため、挿入された列の属性を明示的に指定する必要があります。
SQLセンテンスにおいて最も一般的なインデックスエラーを引き起こす状況に注意を払う必要があります。
例えば、インデックスaのタイプがvarcharであり、SQLセンテンスがwhere a = 1、 varcharをint に変更するなど、タイプを黙示的に変換する場合。
例えば、関数を使用して日付列をフォーマット処理するなど、インデックス列に対して数学的計算と関数等の操作を実行する場合。
join列の文字セットが統一されていない場合。
例えば、インデックスが(a,b)であり 、SQLセンテンスがorder by a b desclikeであるなど、複数の列の並べ替え順序が一致しない場合。
あいまい検索を使用する際、文字タイプ `xxx%` の形式で一部のインデックスを作成できるが、他の状況ではいずれもインデックスを作成できない場合。
否定的なクエリー(not、!=、not inなど) が使用される場合。

推奨事項

必要に応じてリクエストし、 `select *` を拒否して、次の問題を回避します。
インデックスのカバリング、テーブルのリターン操作、 I/Oの追加ができない。
メモリ負荷の増加、大量のコールドデータの `innodb_buffer_pool_size` への流入、クエリーヒット率の低下。

ネットワーク伝送のオーバーヘッドの増加。
大規模トランザクションを極力避け、大規模トランザクションを小さなトランザクションに分割し、マスター/スレーブの遅延を回避します。
業務コード内のトランザクションを速やかに行い、不要なロック待機を回避します。
複数テーブルではjoinの使用を少なくし、大きなテーブルではjoinを禁止します。2つのテーブルのjoinは小さなテーブルをドライバーテーブルとする必要があり、join列は文字セットが一致し、かつすべてにインデックスが作成されている必要があります。
LIMITページングの最適化、LIMIT 80000、10といった操作は、80010件のレコードを取り出してから、後の10件のレコードを返すため、データベースの負荷が極めて高くなります。したがって、例えば `SELECT * FROM test WHERE id >= ( SELECT sql_no_cache id FROM test order by id LIMIT 80000,1 ) LIMIT 10 ;` のように、ページングの前に最初のレコードの位置を確認することをお勧めします。
マルチレベルのサブクエリーが埋め込まれたSQLセンテンスを回避します。MySQL 5.5より前のクエリーオプティマイザはinをexistsに変更するため、インデックスを失効させる恐れがあり、外部テーブルが非常に大きい場合はパフォーマンスが低下します。
説明：
上述の状況を完全に回避することは困難であり、推奨されるソリューションは、これらの条件を主要なフィルタリング条件としないことです。インデックスを作成する際の主要なフィルタリング条件に従えば、大きな問題はありません。
モニタリングでフルテーブルスキャンの量がかなり多いことが判明しました。コンソールパラメータで `log_queries_not_using_indexes` を設定し、しばらくすると低速ログファイル解析をダウンロードすることができますが、低速ログの急増を避けるため、長時間開いたままにしないでください。
業務のオンライン化前に、必要なSQL監査を実行します。日常の運用保守では、対象を絞った最適化を実施するため、スロークエリーログを定期的にダウンロードする必要があります。

アプリケーションの自動再接続機能のコンフィグレーション

最終更新日：：2024-07-25 17:56:18

このドキュメントでは、インスタンス切替時に発生する数秒ぐらいの接続中断の影響と、自動再接続機能の設定について説明します。

背景

MySQLは データベースインスタンス規格の調整、データベースエンジンバーションのアップグレードなどの処理を実行する時、またはマスターインスタンスが高負荷でハングしたり、ハードウェアに故障が発生したりする時など、インスタンスの切替につながる可能性があり、インスタンスの切替時に数秒ぐらいの接続中断が発生することがあります。
アプリケーションに自動再接続機能が設定されていない場合は、マスター/スレーブの切替後にアプリケーションの接続に異常が発生することで、サービスへの通常のアクセスに影響を与えることがあります。
アプリケーションの自動再接続機能を設定し、メンテナンス時間内にインスタンスを切り替えるようにお勧めします。

自動再接続機能のコンフィグレーション

マスター/スレーブの切替によるアプリケーションの接続異常を回避するため、MySQLのアプリケーション自動再接続機能を設定し、接続プールのconnectTimeOutおよびsocketTimeOutパラメータを設定することをお勧めします。
サービスケースごとに適切なパラメータ値を設定し、OLTP（On-Line Transaction Processing）に基づくサービスケースの場合は、すべて20秒を設定することをお勧めします。
説明：
connectTimeOut：データベースサーバにTCP接続を確立するアプリケーションのタイムアウト時間について、少なくともデータベースサーバに対するアプリケーションの応答時間よりも長くすることをお勧めします。
socketTimeOut：TCP接続を介してデータパケットを送信した後に応答を待機するタイムアウト時間について、1つのSQLの最大実行時間に設定することをお勧めします。

MySQLマスターインスタンスパラメータの変更影響

最終更新日：：2024-07-25 17:56:18

MySQLインスタンスの場合は、コンソールでマスターインスタンスのパラメータを変更できます。その中の重要なパラメータについて、不適切な変更方式を使用すると、災害復旧インスタンスの異常やデータの不整合を引き起こす可能性があります。このドキュメントでは、次のように重要なパラメータの変更による影響を説明します。

lower_case_table_names

デフォルト値：0
役割：データベースとテーブルを作成する時、保存と問い合わせの時に大文字と小文字が区別されるか。このパラメータには数値0、1を設定できます。デフォルト値0は、データベースとテーブルの作成時に保存と問い合わせの両方で大文字と小文字が区別されることを意味し、逆の場合は区別されません。
影響：マスターインスタンスのパラメータが変更されると、災害復旧インスタンスのパラメータを同期的に変更できません。マスターインスタンスの大文字と小文字が区別されるが、災害復旧インスタンスの大文字と小文字が区別されない場合、例えば、マスターインスタンスがTestとTEstのテーブル名を持つ2つのテーブルを作成し、災害復旧インスタンスがそのログを適用すると、TEstのテーブル名がすでに存在するため、データの同期ステータスの異常が発生します。

auto_increment_increment

デフォルト値：1
役割：は、自動採番列AUTO_INCREMENTに使用される増分値です。このパラメータは1～65535の範囲で設定でき、デフォルト値は1です。
影響：マスターインスタンスのパラメータが変更されると、災害復旧インスタンスのパラメータを同期的に変更できません。マスターインスタンスが自動採番列の増分値を変更したが、災害復旧インスタンスが同期に変更されない場合、マスターインスタンスと災害復旧インスタンスのデータが一致しない可能性があります。

auto_increment_offset

デフォルト値：1
役割：自動採番列AUTO_INCREMENTに使用される初期値（オフセット値）です。このパラメータは1～65535の範囲で設定でき、デフォルト値は1です。
影響：マスターインスタンスのパラメータが変更されると、災害復旧インスタンスのパラメータを同期的に変更できません。マスターインスタンスが自動採番列の初期値を変更したが、災害復旧インスタンスが同期に変更されない場合、マスターインスタンスと災害復旧インスタンスのデータが一致しない可能性があります。

sql_mode

デフォルト値：NO_ENGINE_SUBSTITUTION
役割：MySQLは別のsqlモードで実行できます。sqlモードでは、mysqlがサポートすべきのsql構文、データ検証などを定義しています。バージョン5.6のデフォルトのパラメータ値はNO_ENGINE_SUBSTITUTIONであり、使用するストレージエンジンが無効またはコンパイルされていない場合にエラーがスローされることを示します。バージョン5.7のデフォルトのパラメータ値
は `ONLY_FULL_GROUP_BY` 、 `STRICT_TRANS_TABLES` 、 `NO_ZERO_IN_DATE` 、 `NO_ZERO_DATE` です。
, `ERROR_FOR_DIVISION_BY_ZERO` , `NO_AUTO_CREATE_USER` , `NO_ENGINE_SUBSTITUTION` ,
そのうち：
`ONLY_FULL_GROUP_BY` はGROUP BYの集約処理時に、SELECT文の列、HAVING或いはORDER BY句の列はGROUP BYに存在するか、GROUP BY列に依存する関数列でなければならないことを示します。
`STRICT_TRANS_TABLES` はStrictモードが有効になっていることを示します。
`NO_ZERO_IN_DATE` は日付の月と日に0が含まれることを許可するか、且つStrictモードを有効にすると影響を受けるかを示しています。
`NO_ZERO_DATE` はデータベースにゼロ日付をインサートすることを許可せず、且つStrictモードを有効にすると影響を受けることを示しています。
`ERROR_FOR_DIVISION_BY_ZERO` は、Strictモードの場合、INSERT或いはUPDATEを実行する時、データがゼロで割ると、警告ではなくエラーが発生するが、Strictモード以外の場合、データがゼロで割ると、MySQLがNULLを返すことを示しています。
`NO_AUTO_CREATE_USER` は、パスワードがNULLのユーザを新規作成するGRANT文が禁止されることを示しています。
`NO_ENGINE_SUBSTITUTION` は、使用するストレージエンジンが無効になっているか、コンパイルされていない場合にエラーがスローされることを示しています。
影響：マスターインスタンスのパラメータが変更されると、災害復旧インスタンスのパラメータを同期的に変更できません。マスターインスタンスがsql modeを変更したが、災害復旧インスタンスが同期に変更されない場合、例えば、マスターインスタンスのsql modeの制限が災害復旧インスタンスのsql mode制限よりも低いことが、マスターインスタンスで実行したSQLが災害復旧インスタンスに同期される時にエラーが発生し、マスターインスタンスと災害復旧インスタンスのデータが一致しない可能性があります。

MyISAMからInnoDBエンジンへの切り替え制限

最終更新日：：2024-07-25 17:56:18

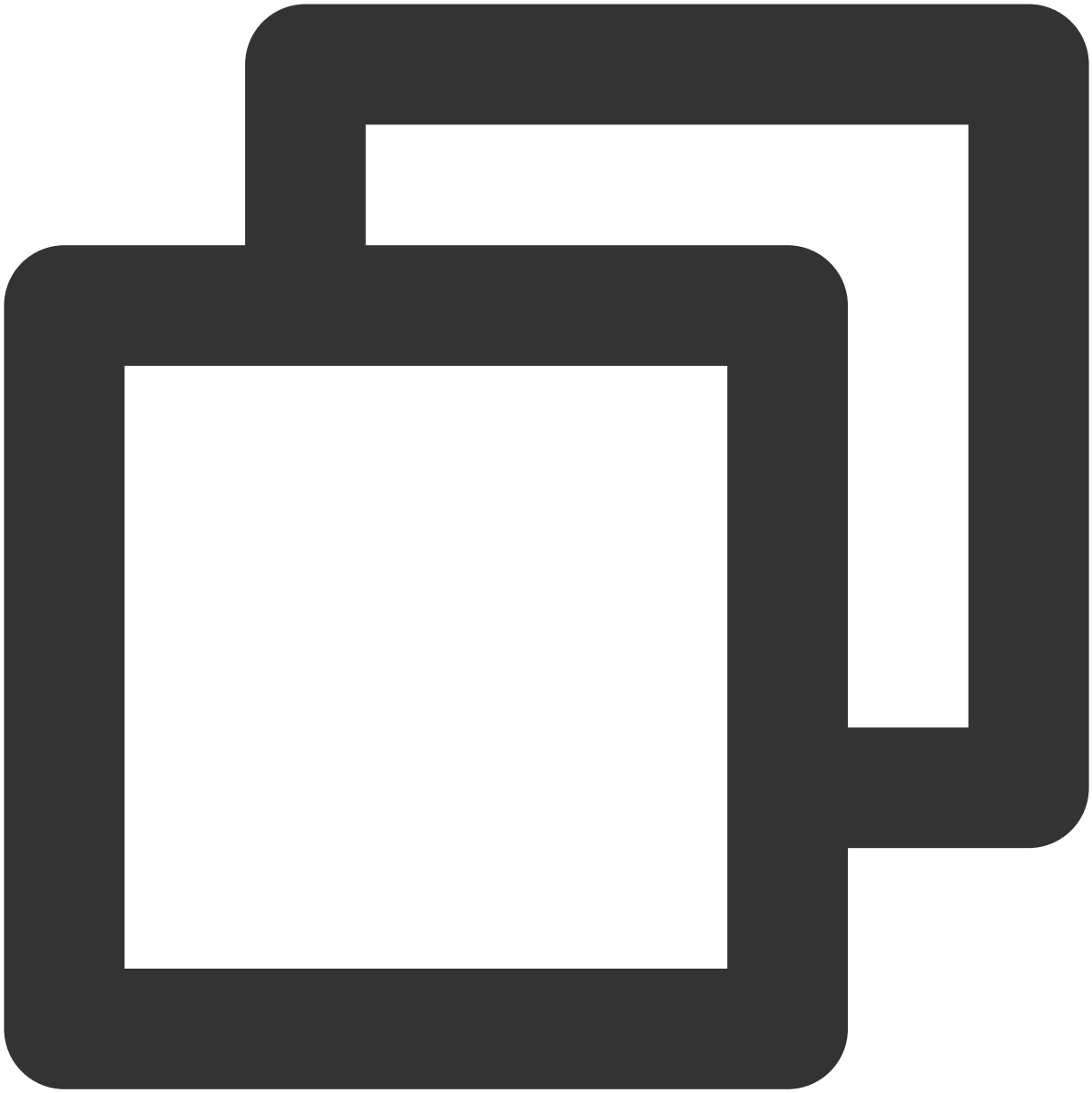
このドキュメントでは、MyISAMエンジンが自動的にInnoDBエンジンに変換された後にテーブルを作成する時にエラーが発生することに対応するソリューションについて説明します。

背景

Tencent MySQLはInnoDBストレージエンジンをデフォルトでサポートしますが、MySQL 5.6以降のバージョンはMyISAMエンジンおよびMemoryエンジンがサポートされていません。詳細については、データベースストレージエンジンをご参照ください。
MySQL 5.6以降のバーションにマイグレーションを行うか、アップグレードすると、システムは自動的にMyISAMエンジンからInnoDBエンジンに変換されます。
MyISAMエンジンは自動採番列を含む複合プライマリキーをサポートしているが、InnoDBエンジンはサポートしていないので、MyISAMエンジンがInnoDBエンジンに変換された後、テーブルの作成時にエラーが発生します。この場合、エラーメッセージは `ERROR 1075 (42000):Incorrect table definition;there can be only one auto column and it must be defined as a key` です。
自動採番列へのインデックス作成を通じてInnoDBエンジンの複合プライマリキーに自動採番列構文を入れることをお勧めします。

InnoDBエンジンの複合プライマリキーに自動採番列が含まれるための変更プログラム

1. テーブルを作成する時にエラーが発生する場合のSQL文：

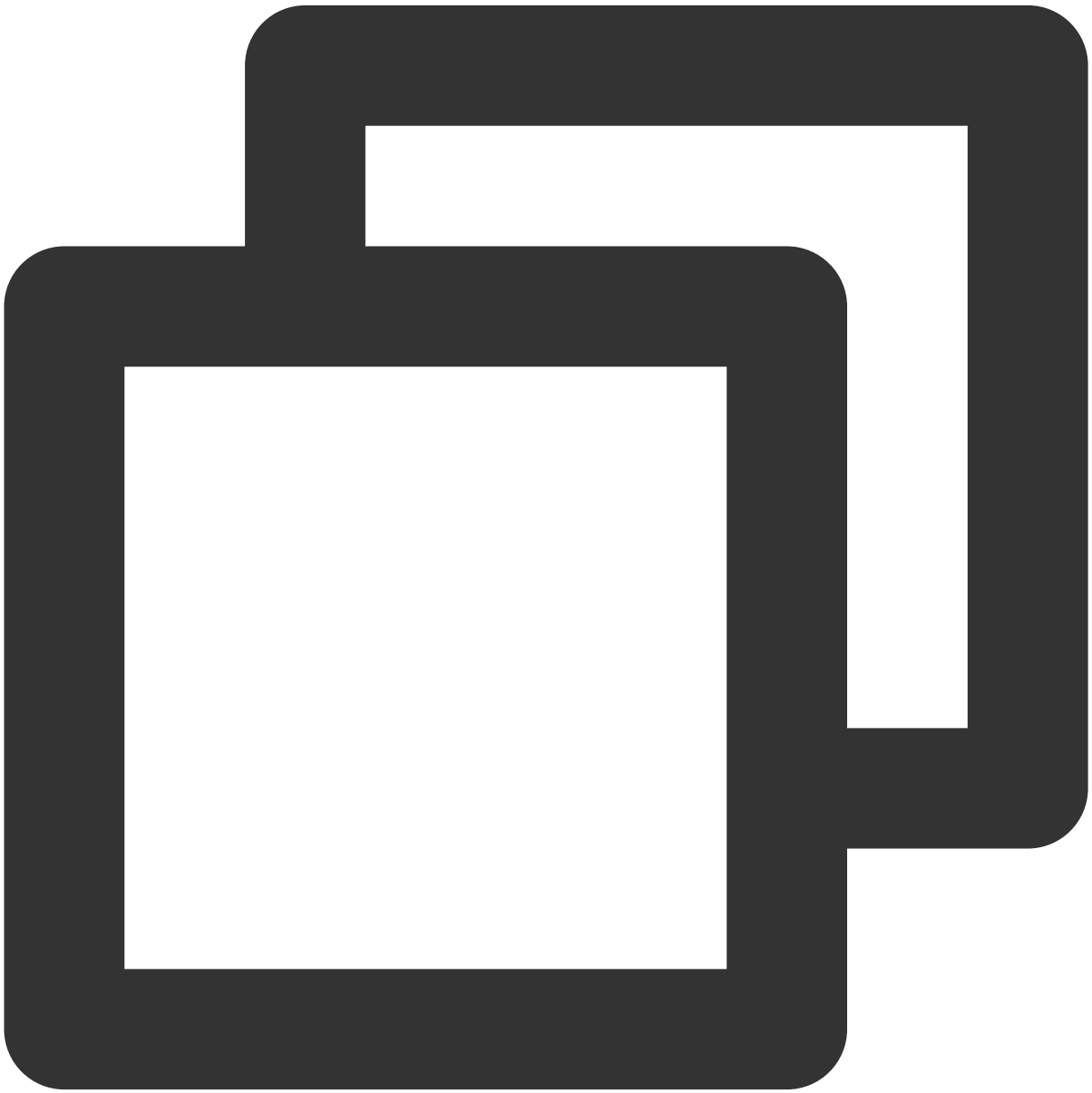


```
create table t_complexkey
(
id int(8) AUTO_INCREMENT,
name varchar(19),
value varchar(10),
primary key (name,id)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

次の図にように作成するとエラーが発生します。

```
MySQL [    ı]> 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
```

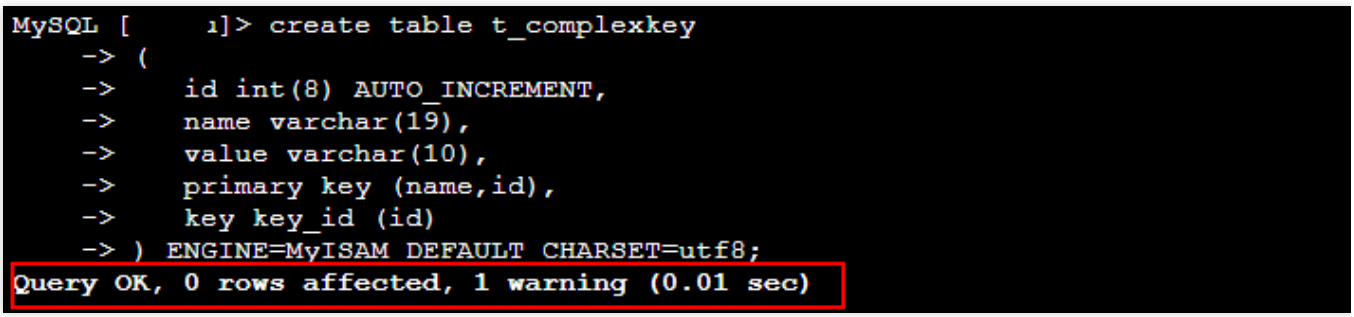
2. インデックス作成後のSQL文を変更します。



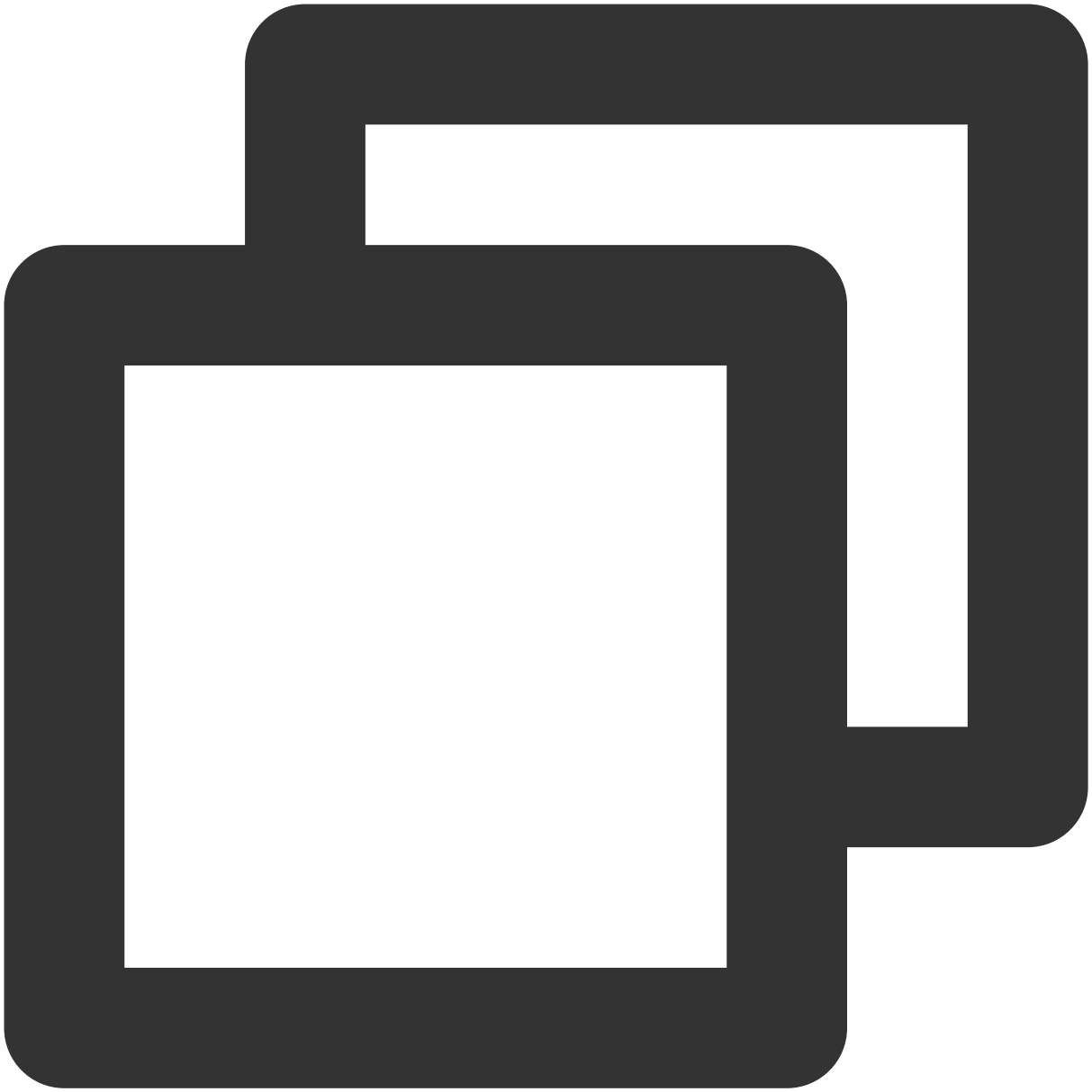
```
create table t_complexkey
```

```
(
id int(8) AUTO_INCREMENT,
name varchar(19),
value varchar(10),
primary key (name,id),
key key_id (id)  ## 自動採番列にインデックスを作成します
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

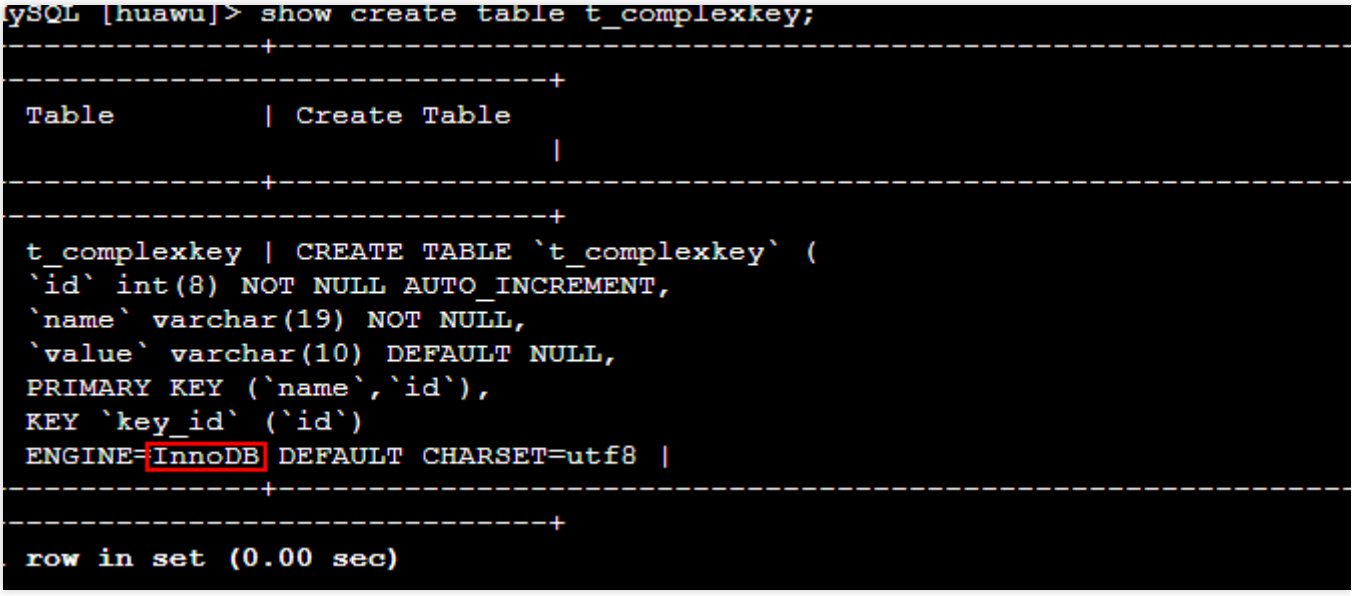
次の図にように作成が完了します。



3. 新規作成されたテーブル構造を確認します。



```
show create table t_complexkey;
```



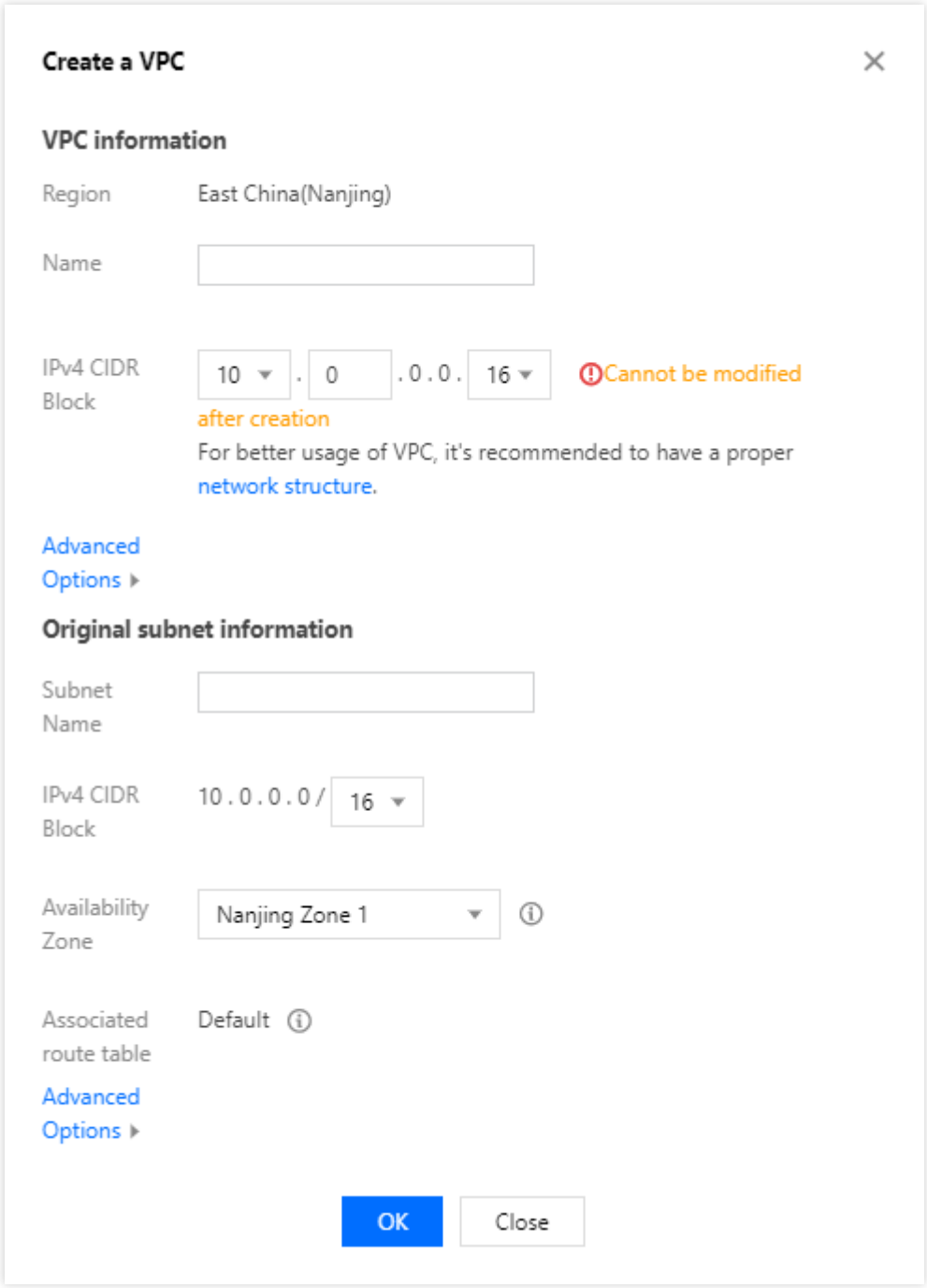
TencentDB for MySQLのためのVPC作成

最終更新日：：2024-07-25 17:56:18

Tencent Cloudは、TencentDBをホスティングするフラットフォーム、Tencent Cloud Virtual Private Cloud（VPC）をご用意しています。これにより、VPCの中でTencent Cloudのリソース（例えば、TencentDBインスタンス）を起動することができます。
同じVPCで実行されているクラウドデータベースインスタンスをWebサーバーとデータ共有するソリューションがよく用いられています。本マニュアルでは、このソリューションに対してVPCを作成し、また、クラウドデータベースをVPCに追加して併用します。
ここでは、同じVPC内に、Cloud Virtual Machine（CVM）とTencentDB for MySQLを追加し、VPCの中で、クラウドリソースのプライベートネットワーク内で相互通信を実現する方法をご紹介します。

手順1：VPCの作成

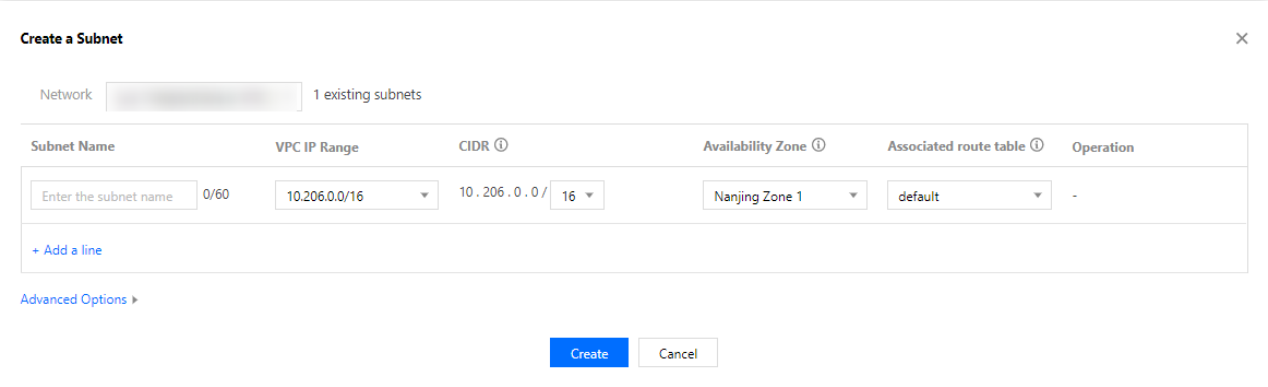
VPCには少なくとも1つのサブネットを含め、サブネットの中にのみクラウドサービスリソースを追加することができます。
1. VPCコンソールにログインします。
2. リストの上方で、VPCが所属するリージョンを選択し、**+新規作成**をクリックします。
3. VPCの情報とオリジナルのサブネットの情報を記入し、**OK**をクリックして完了です。このうち、VPCとサブネットのCIDRは作成後の変更はできません。
VPC CIDRには以下のIPレンジの中のいずれかの使用をサポートしています。異なるVPC間でプライベートネットワーク通信が必要な場合は、両端のCIDRの設定が重複しないようにしてください。
`10.0.0.0` - `10.255.255.255`（マスクの範囲は16 - 28の間）
`172.16.0.0` - `172.31.255.255`（マスクの範囲は16 - 28の間）
`192.168.0.0` - `192.168.255.255`（マスクの範囲は16 - 28の間）
サブネットCIDRはVPC CIDR範囲内または同じにする必要があります。
例えば、VPCのIPレンジが `192.168.0.0/16` の場合、VPC内のサブネットのIPレンジは `192.168.0.0/16` 、 `192.168.0.0/17` などになります。



手順2：サブネットの作成

同時に1つまたは複数のサブネットを作成することができます。

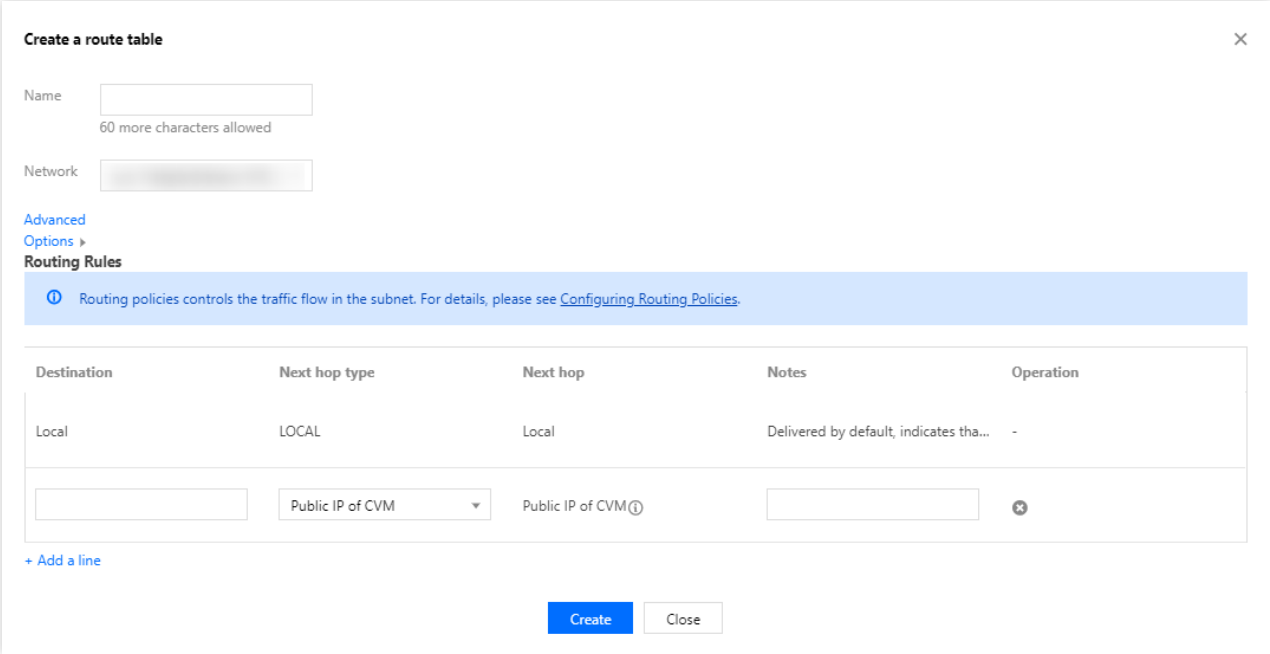
1. VPCコンソールにログインします。
2. 左側のディレクトリの中の**サブネット**をクリックし、管理画面に入ります。
3. サブネットを作成したいリージョンとVPCを選択し、**+新規作成**をクリックします。
4. サブネット名、CIDR、アベイラビリティーゾーン、関連付けるルートテーブルを記入します。



5. **+1行追加**をクリックすると、複数のサブネットを作成できます（任意）。
6. **作成**をクリックすれば完了です。

手順3：ルートテーブルとサブネットとの関連付けの作成

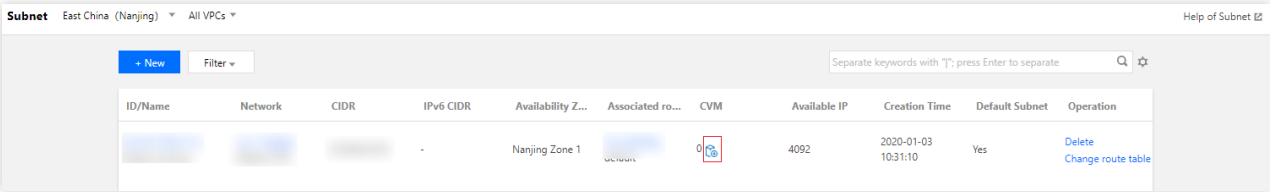
カスタムルートテーブルを作成して、ルーティングポリシーを編集し、その後指定するサブネットに関連付けます。サブネットに関連付けるルートテーブルは、該当サブネットのアウトバウンドルートの指定に用いられます。
1. [VPCコンソール]にログインし、左側の欄から**ルートテーブル**ページを選択します。
2. リストの上方からリージョンとVPCを選択し、**+新規作成**をクリックします。
3. ポップアップしたダイアログボックスの中に名前、所属ネットワーク、新規作成したルーティングポリシーを記入し、**作成**をクリックします。ルートテーブルのリストが返ってきたら新規作成したルートテーブルを確認できます。



4. コンソールの左側の欄から**サブネット**のページを選択して、当該のルートテーブルを関連付けたいサブネットを選択し、**操作**の列の**ルートテーブルの変更**をクリックして、関連付けを行います。

手順4：Cloud Virtual Machine（CVM）の追加

1. VPCコンソールにログインします。
2. 左側のディレクトリの中の**サブネット**をクリックし、管理画面に入ります。
3. CVMを追加したいサブネットの行で、CVMのアイコンをクリックして追加します。

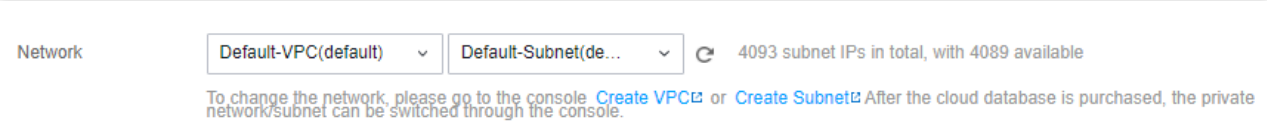


4. 画面の表示にしたがって、CVMを購入すれば完了です。詳細については、CVMのドキュメントの購入方法をご参照ください。

手順5：クラウドデータベースの追加

データベースの新規作成

1. TencentDB for MySQLにログインし、**新規作成**をクリックすると、購入ページに入ります。
2. 購入ページで**ネットワーク**オプションに、**VPC**を選択し、すでに作成したVPCと対応するサブネットを選択すると、新規に購入したクラウドデータベースをVPCに追加します。



既存のデータベース

1. インスタンスリストの中で、インスタンス名または操作の列の**管理**をクリックして、インスタンス詳細画面に入ります。
2. 詳細画面の**所属ネットワーク**で、対応するVPCを切り替えることができます。

MySQLによるサービス負荷能力の向上

最終更新日：：2024-07-25 17:56:18

優れた性能と拡張性を備えたデータベースは、システムの既存の負荷容量の速やかな引き上げをサポートします。同じデータベーススケールでTencentDB for MySQLを適切に使用した場合、データベースの並列能力を高め、業務の1秒あたりのアクセス数の向上を後押しします。

1. 適切なデータベース構成の選択

1.1 データベースバージョンの選択

TencentDB for MySQLは現在、ネイティブMySQLと完全に互換性のあるバージョン5.5、5.6、5.7、8.0を提供し、より安定した使用には5.6またはそれ以降のバージョンを選択することをお勧めします。データベースカーネルは、5.5以前のバージョンの設計を最適化してシステム性能の向上を図り、さまざまな魅力的な新機能を提供しています。

このドキュメントではMySQL 5.7を例に取り、新しいバージョンの機能についてご紹介しています。MySQL 5.7は一般的に、高性能、信頼性、使いやすさを備えていると認識されています。その最適化ポイントと新機能の一部を次に示します：

ネイティブJSONのサポート

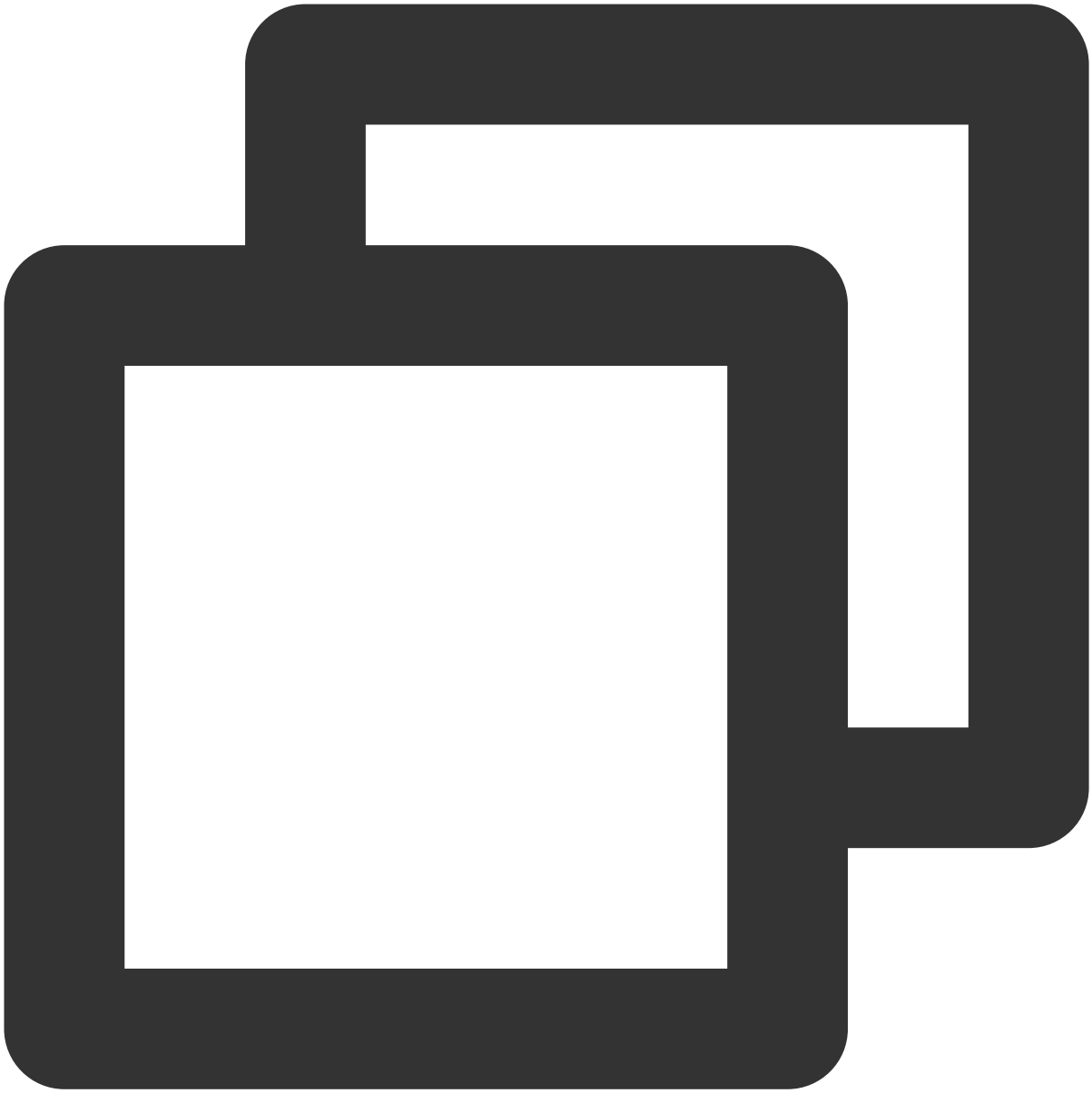
MySQL 5.7には、MySQLテーブルにJSON形式でデータを格納するための新しいデータタイプが追加されています。JSONデータタイプのネイティブサポートの主なメリットを次に示します。

ドキュメントのチェック：JSON仕様に準拠するデータセグメントのみをJSON型の列に書き込むことができるため、自動的にJSON構文チェックが実行されたことになります。

効率的なアクセス：JSON型の列にJSONドキュメントを保存すると、データはプレーンテキストとして保存されず、最適化されたバイナリー形式で保存されるため、オブジェクトのメンバーと配列要素に速やかにアクセスすることができます。

性能の向上：JSON型の列データにインデックスを作成して、queryの性能を向上させることができます。このようなインデックスは、仮想列上に作成された「関数インデックス」によって実現できます。

利便性：JSON型の列に追加されたインライン構文は、自然にSQLステートメントのドキュメントクエリーに統合することができます。例えばfeatures、featureはJSONフィールドです。



```
SELECT feature->"$.properties.STREET" AS property_street FROM features WHERE id =
```

MySQL 5.7を使用すると、最適なリレーショナルパラダイムとドキュメントパラダイムを1つのツールにシームレスに組み合わせることができ、さまざまなアプリケーションやユースケースに最適なリレーショナルパラダイムまたはドキュメントパラダイムを使用することができます。これによって、MySQLユーザーのアプリケーションの範囲が大幅に拡大します。

SYS Schema

MySQL SYS Schemaは、一連のオブジェクト（ビュー、ストアドプロシージャ、ストレージメソッド、テーブルおよびトリガー）で構成されるdatabase schemaです。これにより、Performance Schemaや

INFORMATION_SCHEMAのさまざまなテーブルに格納されている監視データリソースに、手軽かつ読み取り可能、さらにDBAと開発者に配慮した方法でアクセスすることができます。
MySQL SYS Schemaはデフォルトで MySQL5.7に含まれており、次のよくある質問に答えるための要約ビューを提供します。
データベースサービスのすべてのリソースを占有しているのは何ですか？
データベースサーバーへのアクセスが最も多いホストはどれですか？
インスタンスのメモリはどこで使用されますか？
InnoDBの改善
InnoDBオンライン操作（Online DDL）：MySQLを再起動せずに、お客様のBuffer Pool sizeを動的に調整して、ニーズの変化に対応することができます。今ではInnoDBも、InnoDBのUNDOログとテーブルキャパシティをオンラインで自動的にクリアにして、共有テーブルキャパシティファイル（ibdata1）が大きくなる問題が生じる一般的な原因の1つが解消されます。最後に、過去バージョンではインデックスまたはテーブルを再構築する必要がありましたが、MySQL 5.7はインデックスの名前変更とvarcharのサイズの変更をサポートしています。
InnoDBネイティブパーティション：MySQL 5.7 InnoDBには、パーティションのネイティブサポートが含まれています。InnoDBネイティブパーティションにより負荷が軽減され、メモリ要件を最大90%低下させます。
InnoDBキャッシュのウォームアップ：MySQLがリスタートすると、InnoDB はバッファープール内の最もホットなデータの25%を自動的に保持します。そのため、データキャッシュのプリロードやウォームアップは不要になり、MySQLの再起動による性能の低下が生じることもありません。
MySQL 5.7のその他の最適化と新機能については、MySQL公式データをご参照ください。

1.2インスタンス仕様の選択（データベースメモリ）

MySQLは現在、個別のCPUオプションを提供していません。CPUはメモリの仕様に従い、割合に応じてアサインされます。お客様はご自分の業務の特性に応じて、対応するデータベース仕様を購入することができます。モデルを選択する際の性能の参考を提供するために、インスタンスごとに詳細な標準化テストを行っています。
ただし、Sysbenchの性能テストは、すべての業務シナリオを表しているわけではないことにご注意ください。業務シナリオでのMySQLの性能やパフォーマンスをよりよく理解するために、MySQLで本格的に業務を実行する前に、データベースで負荷テストを実行することをお勧めします。MySQL性能の説明をご参照ください。
メモリはインスタンスのコアメトリックの1つであり、アクセス速度はディスクよりもはるかに高速です。通常はメモリキャッシュのデータが多いほど、データベースの応答は速くなります。メモリが小さい場合にデータが一定量を超えると、ディスクにフラッシュされます。新しいリクエストがこのデータに再度アクセスする場合、ディスクからメモリに読み込む必要があり、ディスクIOを消費します。この際、データベースの応答が遅くなります。
**読み取りの並列性が大きい、または読み取り遅延にセンシティブな業務の場合、データベースの性能を確保するため、小さすぎるメモリ仕様は選択しないことをお勧めします。 **

1.3 ディスクの選択

TencentDB for MySQLインスタンスのディスクキャパシティには、データファイル、システムファイル、binlogファイル、一時ファイルが含まれます。書き込まれるデータ量がインスタンスのディスクキャパシティを超える場合、速やかにアップグレードしないとインスタンスロックがトリガーされる可能性があります。従って、ディスク

キャパシティを購入するときは、ディスク容量の不足によるインスタンスのロックや頻繁なアップグレードを回避するために、後日のデータ増加に備えてある程度の冗長性を持たせることをお勧めします。

1.4 適切なデータレプリケーション方法の選択

TencentDB for MySQLは、非同期、半同期、および強い同期という3つのレプリケーションモードを提供しています。データベースインスタンス・レプリケーションモードをご参照ください。お客様の業務が書き込みレイテンシーやデータベースの性能にセンシティブである場合、非同期レプリケーションを選択することをお勧めします。

1.5 TencentDBの高可用性

TencentDB for MySQLの高可用性は、マスター／スレーブ（M-S）（アクティブ・スタンバイモード）アーキテクチャによって保証されています。マスター／スレーブのデータ同期は、binlogログ方式に依存しています。同時に、インスタンスの任意のタイミングへのロールバックをサポートしています。この機能は、バックアップとログの使用に依存しています。従って通常は、インスタンスの高可用性を確保するためにマスター／スレーブシステムを構築したり、その他の追加費用を支払ったりする必要はありません。

1.6 TencentDBの拡張性

TencentDB for MySQLのデータベースバージョン、メモリやディスクの仕様は、オンラインの動的なホットアップグレードをサポートしています。アップグレードプロセスによって業務が中断されることはなく、業務の規模拡大に伴うデータベースのボトルネックを心配する必要もありません。

1.7 CVMとMySQLの連携使用

通常、購入した後は、クラウドサーバーCVMとMySQLを連携させて使用する必要があります。 CVMを使用したMySQLへのアクセスをご参照ください。

2. 読み取り専用インスタンスを使用した読み取り拡張

一般的なインターネットビジネスでは、データベースの読み取りと書き込みの比率は通常4：1から10：1の間です。このタイプの業務シナリオでは、データベースの読み取り負荷は書き込み負荷よりもはるかに大きくなります。性能ボトルネックが発生した場合の一般的なソリューションは、読み取り負荷を増やすことです。
MySQL読み取り専用インスタンスは、読み取り拡張ソリューションを提供します。読み取り専用インスタンスをご参照ください。
読み取り専用インスタンスは、さまざまなサービスの読み取り専用アクセスにも応用できます。例えばマスターインスタンスは、オンライン業務向けの読み取り／書込みアクセスを行い、読み取り専用スレーブインスタンスは内部業務またはデータ分析プラットフォーム向けに読み取り専用クエリを提供します。

3. クラウドデータベースのディザスタリカバリソリューション

TencentDB for MySQLは、ディザスタリカバリインスタンスを提供し、ワンクリックでリージョン間リモートデータベースのディザスタリカバリの構築をお手伝いします。
ディザスタリカバリインスタンスを使用すると、複数のリージョン間の各コンピュータルームの相互の冗長化が実現できます。1つのコンピュータルームに障害が発生したり、不可抗力事由によりサービスを提供できなくなったりすると、すぐに別のコンピュータルームに切り替えることができます。ディザスタリカバリインスタンスは、データ同期にTencentのプライベートネットワーク専用回線を使用します。さらにMySQLのカーネルレベルのレプリケーションによって最適化されており、災害時の同期遅延による業務への影響を可能な限り排除します。リモートビジネスロジックの準備が整うと、秒単位でのディザスタリカバリの切り替えが可能になります。

4. 2地域3センターソリューション

TencentDB for MySQLを使用すれば、画面上で簡単な設定をいくつか行うだけで、2地域3センターソリューションを実現することができます。
MySQLと同一都市の強整合性クラスターを購入し、マルチアベイラビリティーゾーンのデプロイ（カナリアリリース）を選択すると、1地域2センターの機能が提供されます。
リモートディザスタリカバリノードをこのクラスターに追加すれば、2地域3センターアーキテクチャが実装できます。

5. ディザスタリカバリインスタンスを使用して、ユーザーに最寄りサーバーへのアクセスを提供

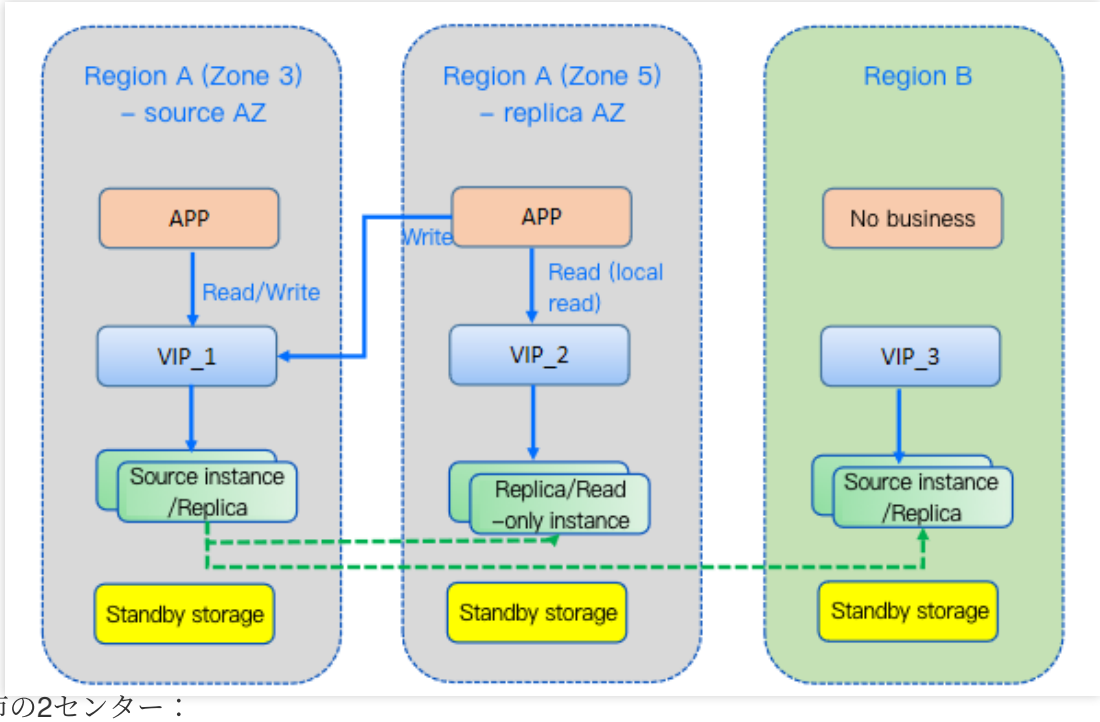
ディザスタリカバリインスタンスは、独立したM-S（プライマリ/スタンバイモード）の高可用性アーキテクチャを採用するとともに、外部へ読み取り専用アクセス機能を提供できます。従って、複数リージョンにまたがる必要のあるユーザーの最寄りサーバーへのアクセスを行うビジネスシナリオの場合、ディザスタリカバリインスタンスを安心して使用することができます。

2地域3センターのディザスタリカバリ構築

最終更新日：：2024-07-25 17:56:18

このドキュメントは、アベイラビリティーゾーン間のインスタンスデプロイとリモートでのディザスタリカバリインスタンスの構築により、2リージョン3センターのアーキテクチャ構築を実現することについてご説明します。

2リージョン3センターのアーキテクチャ構築



同都市の2センター：

リージョンAの3区とリージョンAの5区から同都市の2センターを構成し、データベースを保護するためにリージョンAの3区がサーバーダウンの後にリージョンAの5区に切り替えることができます。

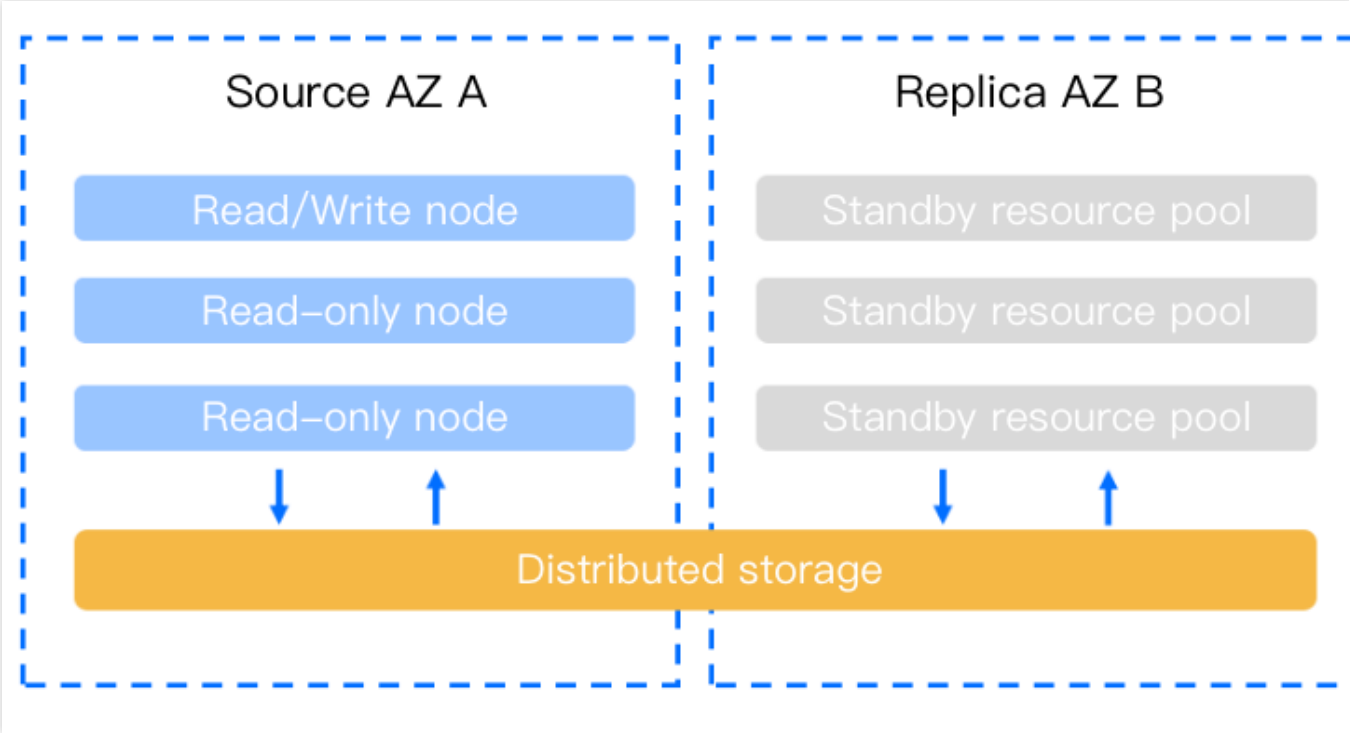
2リージョン：

リージョンAとリージョンBから2地域を構成し、リージョンAの3区、5区のデータセンターがすべてダウンした後、業務もリージョンBのデータセンターに切り替えることができます。

アベイラビリティーゾーン間のデプロイ

TencentDB for MySQLは、アベイラビリティーゾーン間のデプロイをサポートし、シングルアベイラビリティーゾーンとマルチアベイラビリティーゾーンのデプロイ方式よりも高いディザーストリカバリー能力を備え、データベースを保護することで、データベースインスタンスに障害が発生したり、アベイラビリティーゾーンが中断されたりすると、データセンターレベルの障害から保護されます。

TencentDB for MySQLのマルチアベイラビリティーゾーンデプロイにより、データベースインスタンスのために高可用性及びフェイルオーバーのサポートを提供しています。マルチアベイラビリティーゾーンは単一利用可能ゾーンの上、同じリージョンの複数の単一利用可能ゾーンを物理的ゾーンに組み合わせることです。



前提条件

インスタンスステータスが実行中であること。
インスタンスのあるリージョンに2つ以上のアベイラビリティーゾーンが含まれること。
デプロイするアベイラビリティーゾーンが十分のコンピューティングリソースがあること。

サポートするリージョンとアベイラビリティーゾーン

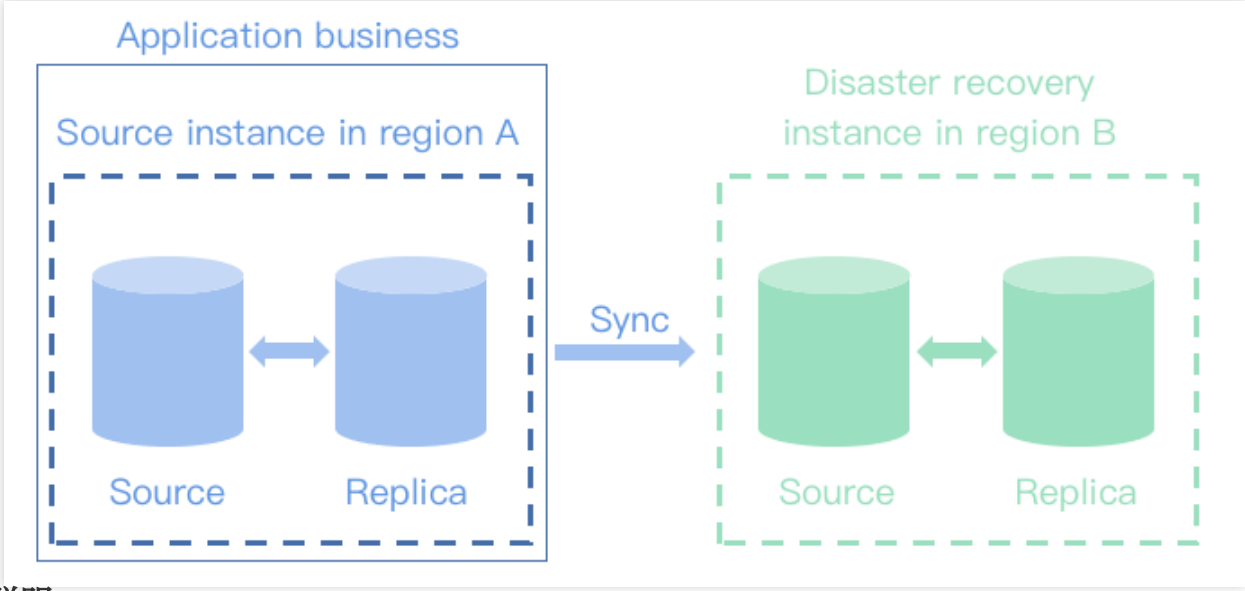
TencentDB for MySQLのマルチアベイラビリティーゾーンのデプロイは、現在広州、上海、南京、北京、成都、重慶、香港、シンガポール、ジャカルタ、バンコク、ムンバイ、ソウル、東京、バージニア、フランクフルトをサポートします。対応する都市がサポートするマスター/スレーブのアベイラビリティーゾーンの選択範囲は、TencentDB for MySQLの購入ページのマスター/スレーブのアベイラビリティーゾーンの掲示を基準とします。
この機能についてはサポートするリージョンとアベイラビリティーゾーンの充実が進んでいます。
業務の必要があれば、チケットを提出しで他のリージョンとアベイラビリティーゾーンのデプロイを申請することができます。

料金説明

本機能は無料です。インスタンスをシングルアベイラビリティーゾーンからマルチアベイラビリティーゾーンへ移行したとしても、料金はかかりません。

リモートディザスタリカバリインスタンス

サービスの継続提供、データの信頼性または監視の必要性が高いシナリオの場合、TencentDB for MySQLは、リージョンをまたがるディザスタリカバリインスタンスを提供し、ユーザーの低コストでのサービスの継続提供能力を引き上げながら、データの信頼性を向上させるお手伝いをします。
リモートディザスタリカバリインスタンスが独立したデータベース接続アドレスを提供し、ディザスタリカバリインスタンスは最寄りへのアクセス、データ分析などのシナリオのための読取アクセス機能を提供することができ、デバイスの冗長性コストは低く、プライマリ/セカンダリの高可用性アーキテクチャを使用することにより、データベースの単一障害点リスクを回避します。



料金説明

ディザスタリカバリインスタンスとマスタインスタンスが同じです。

TencentDB for MySQLのインスタンスのために2リージョン3センターアーキテクチャを構築する

手順1：アベイラビリティーゾーン間のデプロイを設定する

インスタンスを購入するときに、インスタンスのためにアベイラビリティーゾーンデプロイを選択する

1. MySQLコンソールにログインし、インスタンスリストで**新規作成**をクリックして、購買ページに進みます。
2. 購入ページで、適切なサポートありリージョンを選択した後、**スレーブアベイラビリティーゾーン**オプションからマスターアベイラビリティーゾーンとは異なるものを選択します。

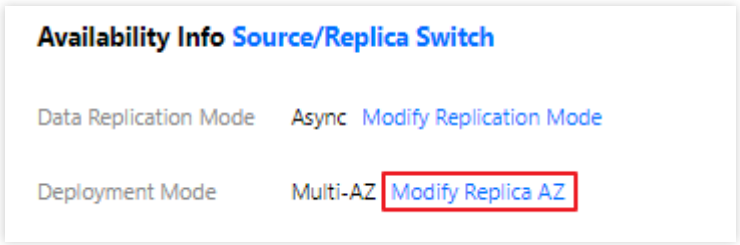
説明：

マスターマシンとスレーブマシンが異なるアベイラビリティーゾーンにある場合、ネットワークの同期遅延が2〜3ms増加する可能性があります。

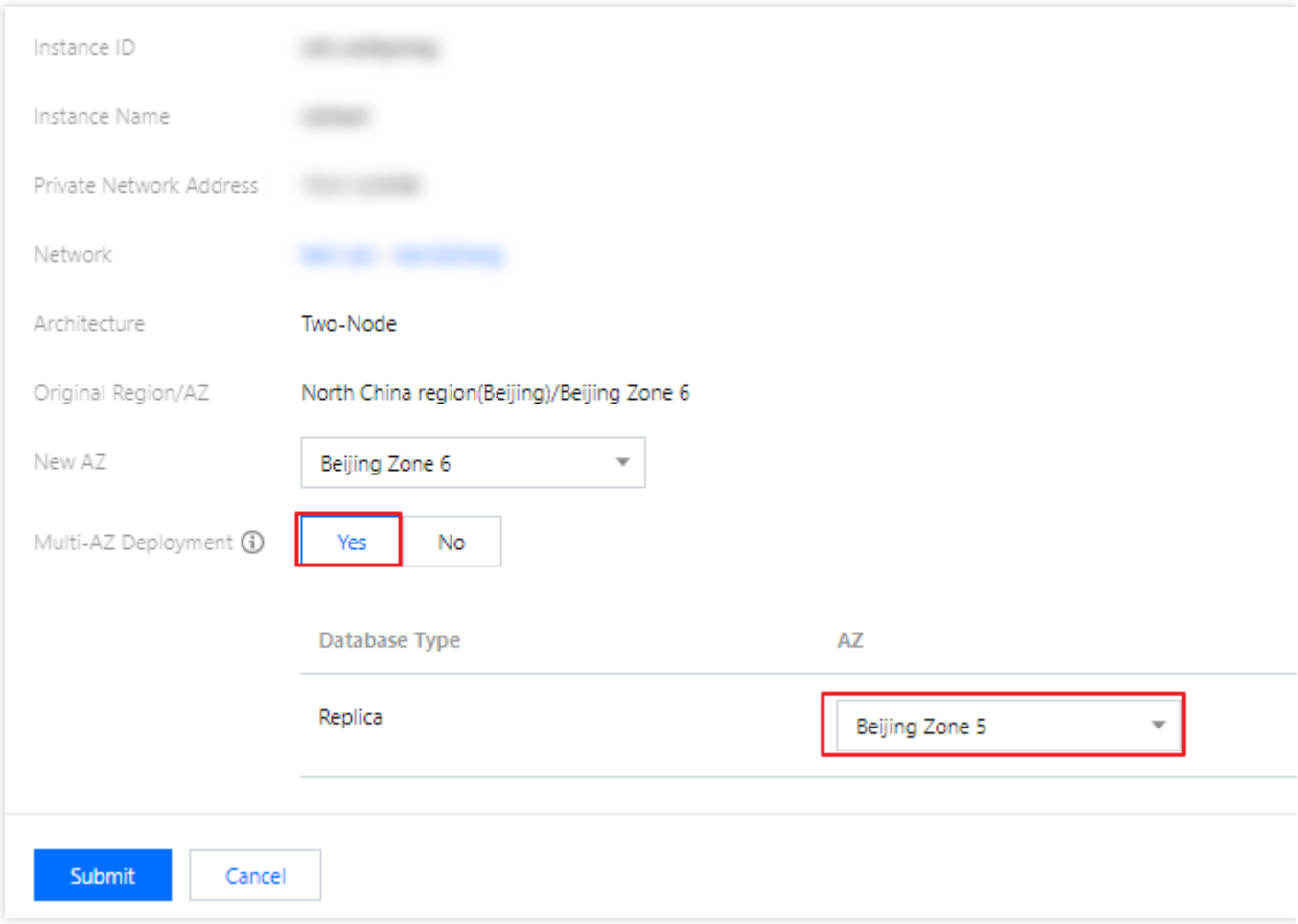
3. 誤りのないことを確認したら、**今すぐ購入**をクリックして、支払いを完了すると、**インスタンス詳細ページ > 可用性情報**でこのインスタンスのマスター/スレーブアベイラビリティーゾーンを確認できます。

インスタンスが作成され、シングルアベイラビリティーゾーンがマルチアベイラビリティーゾーンに変更した

1. MySQL コンソールにログインします。
2. インスタンスリストで、アベイラビリティーゾーンを変更する必要があるインスタンスIDまたは**操作**列の**管理**をクリックし、インスタンス詳細ページに進みます。
3. インスタンス詳細ページの**可用性情報 > デプロイ方式**で、**アベイラビリティーゾーンの変更**をクリックします。



4. ポップアップ表示されたダイアログボックスで、マルチアベイラビリティーゾーンのデプロイを**はい**にして、スタンバイデータベースのアベイラビリティーを選択し、**提出**をクリックします。



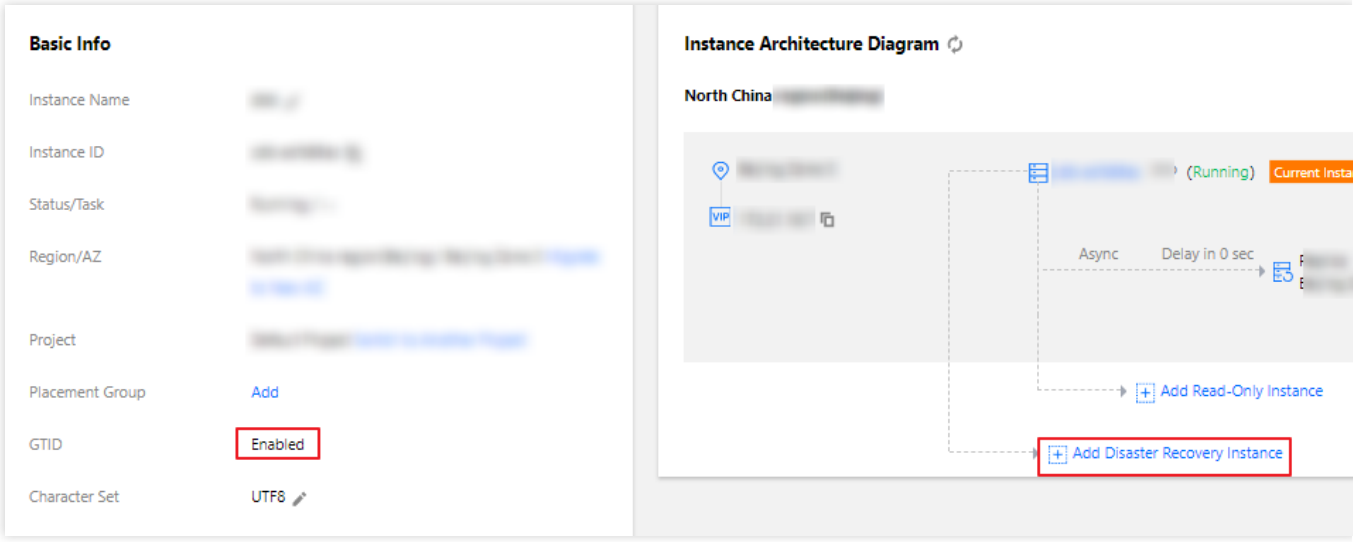
ステップ2：リモートディザスタリカバリインスタンスの設定

説明：

ディザスタリカバリインスタンスを購入するには、メモリ1GB、ハードディスク50GB以上の仕様で、かつMySQL 5.6およびそれ以降のバージョン、InnoDBエンジンの高可用版マスターインスタンスが必要です。マスターインスタンスがこの仕様よりも低い場合は、まずマスターインスタンスの仕様をアップグレードしてください。また、マスターインスタンスでGTID機能が有効になっている必要があります。

2.1 ディザスタリカバリインスタンスの作成

1. MySQLコンソールにログインし、インスタンスリストでインスタンスIDまたは**操作**カラムの**管理**をクリックして、詳細ページに進みます。
2. インスタンス詳細ページの基本情報にて、GTID機能が有効であることを確認し、インスタンスアーキテクチャ図で**ディザスタリカバリインスタンスの追加**をクリックして、ディザスタリカバリインスタンスの購入ページに進みます。



3. 購入ページで、ディザスタリカバリインスタンスの課金モード、リージョン、同期ポリシーなどの基本情報を設定します。

同期ポリシーが**すぐに同期**の場合、データはディザスタリカバリインスタンスが作成された直後に同期されます。

同期ポリシーが**作成後に同期**の場合、ディザスタリカバリインスタンスの作成完了後、ディザスタリカバリ同期リンクを設定してください。操作の詳細については、以下の同期リンクの作成をご参照ください。

説明：

時間の作成はデータ量の影響を受けます。その間のマスターインスタンスのコンソール操作はロックされますので、適切に調整してください。

現在、インスタンス全体のデータ同期のみをサポートしています。ディスク容量を十分に確保してください。

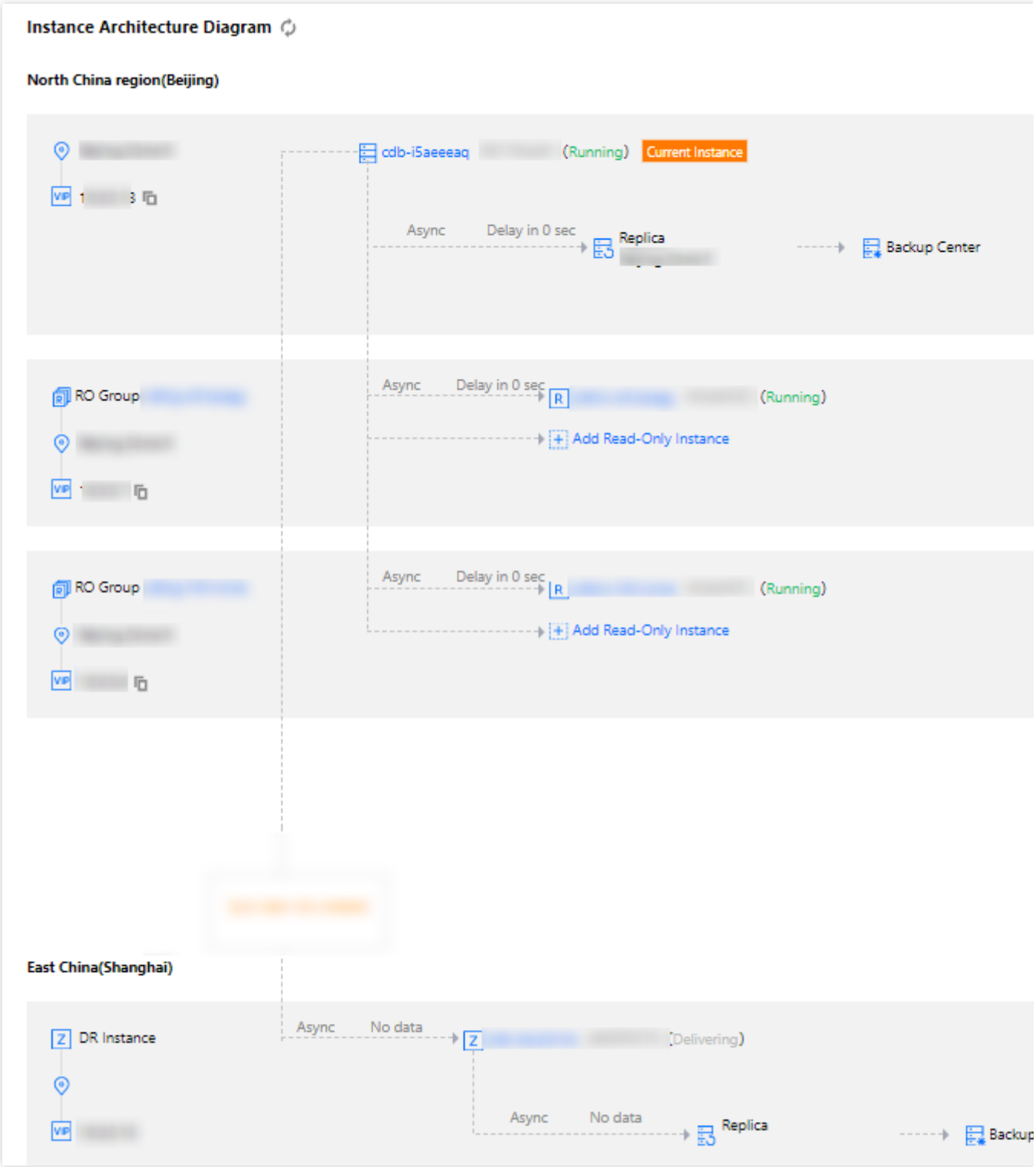
マスターインスタンスのステータスが実行中であり、アップ/ダウングレードや再起動などの実行関連の変更タスクの実行がないことを確認してください。そうでない場合、同期タスクは失敗する可能性があります。

4. 誤りのないことを確認したら、**今すぐ購入**をクリックすると、ディザスタリカバリインスタンスが出荷されます。

5. インスタンスリストに戻り、インスタンスの状態が**実行中**になれば、それ以降の操作に進むことができます。

2リージョン3センターのアーキテクチャの確認

構成完了後、データベースのデプロイ方式は、次のような2リージョン3センターアーキテクチャが表示されます。

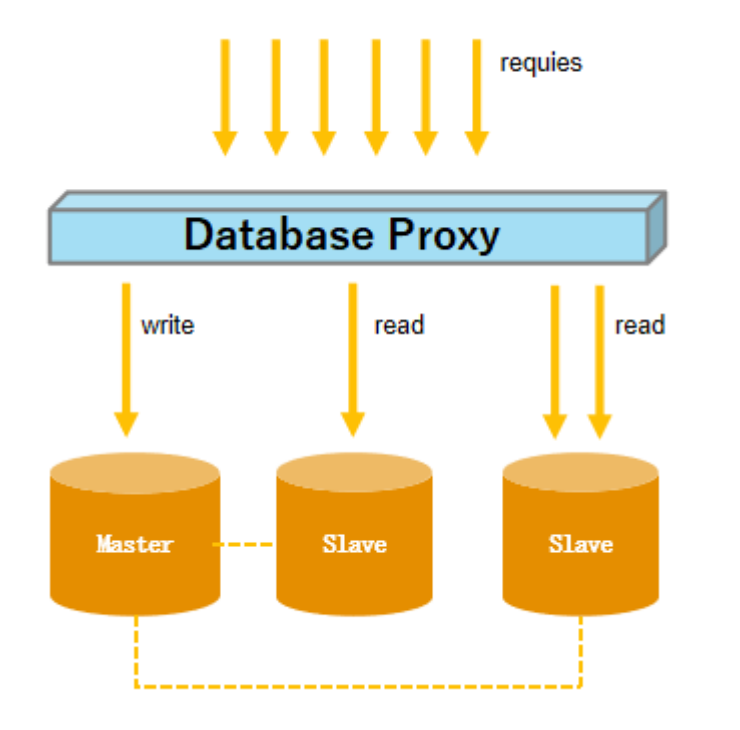

Instance Architecture Diagram
North China region(Beijing)
cdb-i5aeeeaq
(Running)
Current Instance
Async
Delay in 0 sec
Replica
Backup Center
RO Group
Async
Delay in 0 sec
(Running)
Add Read-Only Instance
RO Group
Async
Delay in 0 sec
(Running)
Add Read-Only Instance
East China(Shanghai)
DR Instance
Async
No data
Delivering)
Async
No data
Replica
Backup

リード・ライト分離によるTencentDB for MySQLパフォーマンスの拡張

最終更新日：：2024-07-25 17:56:18

このドキュメントでは、データベースプロキシによりリード・ライト分離機能をオンにすることで、スケールアウトを実現し、TencentDB for MySQLのパフォーマンスを向上させるすることをご紹介します。

データベースプロキシがリード・ライト分離アーキテクチャを実現する



データベースプロキシ

データベースプロキシは、クラウドデータベースサービスとアプリケーションサービスの間にあるネットワークプロキシサービスであり、アプリケーションサービスがデータベースにアクセスするときにすべてのリクエストをプロキシするために用いられます。
データベースプロキシのアクセスアドレスは、元のデータベースのアクセスアドレスから独立しています。データ

ベースプロキシアドレスを介したすべてのリクエストは、プロキシクラスターを介して中継され、データベースのマスターノードとスレーブノードにアクセスし、読み取り/書き込み分離を行います。読み取りリクエストは読み取り専用インスタンスに転送され、メインデータベースへの負荷が軽減されます。

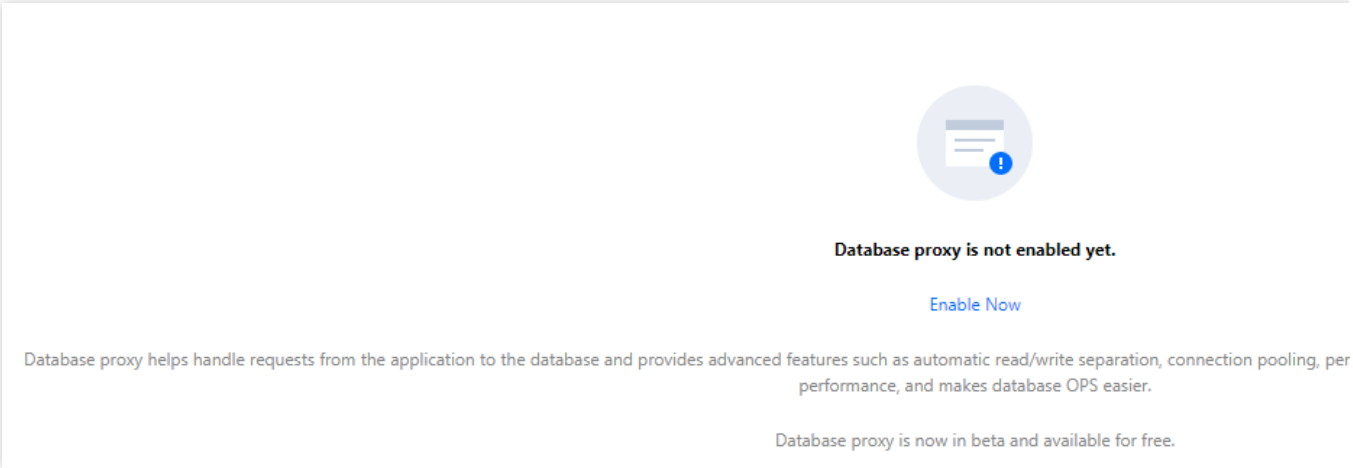
自動リード・ライト分離

ユーザーのビジネスシナリオには、読み取りの増加と書き込みの減少、予測できない業務の負荷といったシナリオがあります。ただし、読み取りリクエスト数が多いアプリケーションシナリオでは、単一のインスタンスが読み取りのプレッシャーに耐えられない場合があり、業務に影響を与える可能性もあります。
読み取り機能のフレキシブルな拡張を実現し、データベースのプレッシャーを分担するために、1つ以上の読み取り専用インスタンスを作成し、読み取り専用インスタンスを利用して多数のデータベース読み取り要件を満たすことができます。ただし、このタイプのソリューションでは、業務側が読み取り/書き込み分離の変換をサポートする必要があります。そのコードのロバスト性によって、ビジネスの読み取り/書き込み分離の品質が決まるため、顧客に対する技術的要件が高く、柔軟性とスケーラビリティが低くなります。
したがって、読み取り専用インスタンスの作成後、データベースプロキシを購入することで読み取り/書き込み分離機能を有効にし、データベースプロキシアドレスをアプリケーションで設定することによって、書き込みリクエストをマスターインスタンスに、読み取りリクエストを各読み取り専用インスタンスに自動的に転送できます。

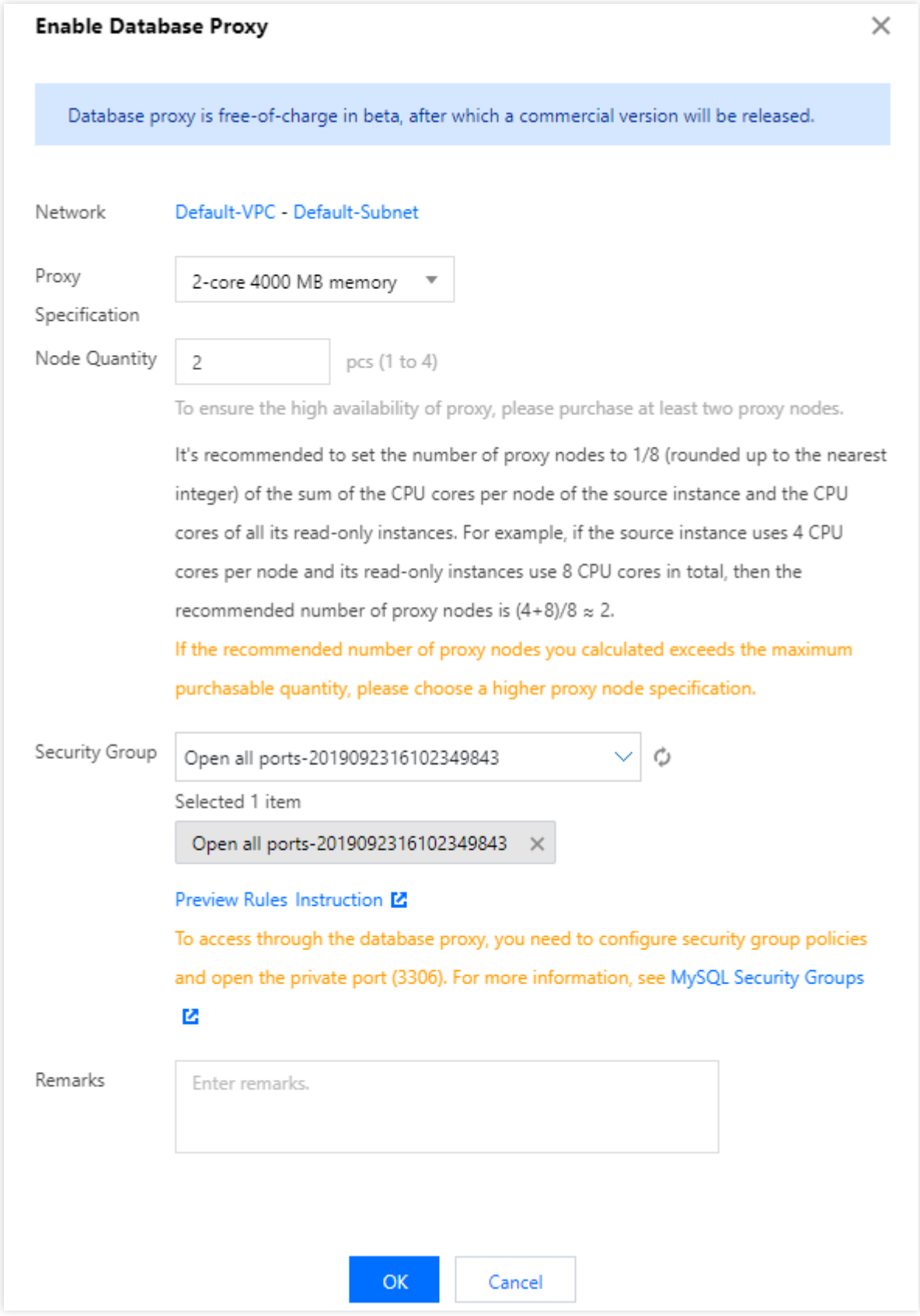
データベースプロキシによりリード・ライト分離機能を有効にする

ステップ1：データベースプロキシを有効にする

1. MySQLコンソールにログインし、インスタンスリストで、プロキシの有効化が必要なマスターインスタンスを選択し、インスタンスIDまたは**操作**の列の**管理**をクリックし、インスタンス管理ページに進みます。
2. インスタンス管理ページで、**データベースプロキシ**ページを選択し、**今すぐ有効化**をクリックします。



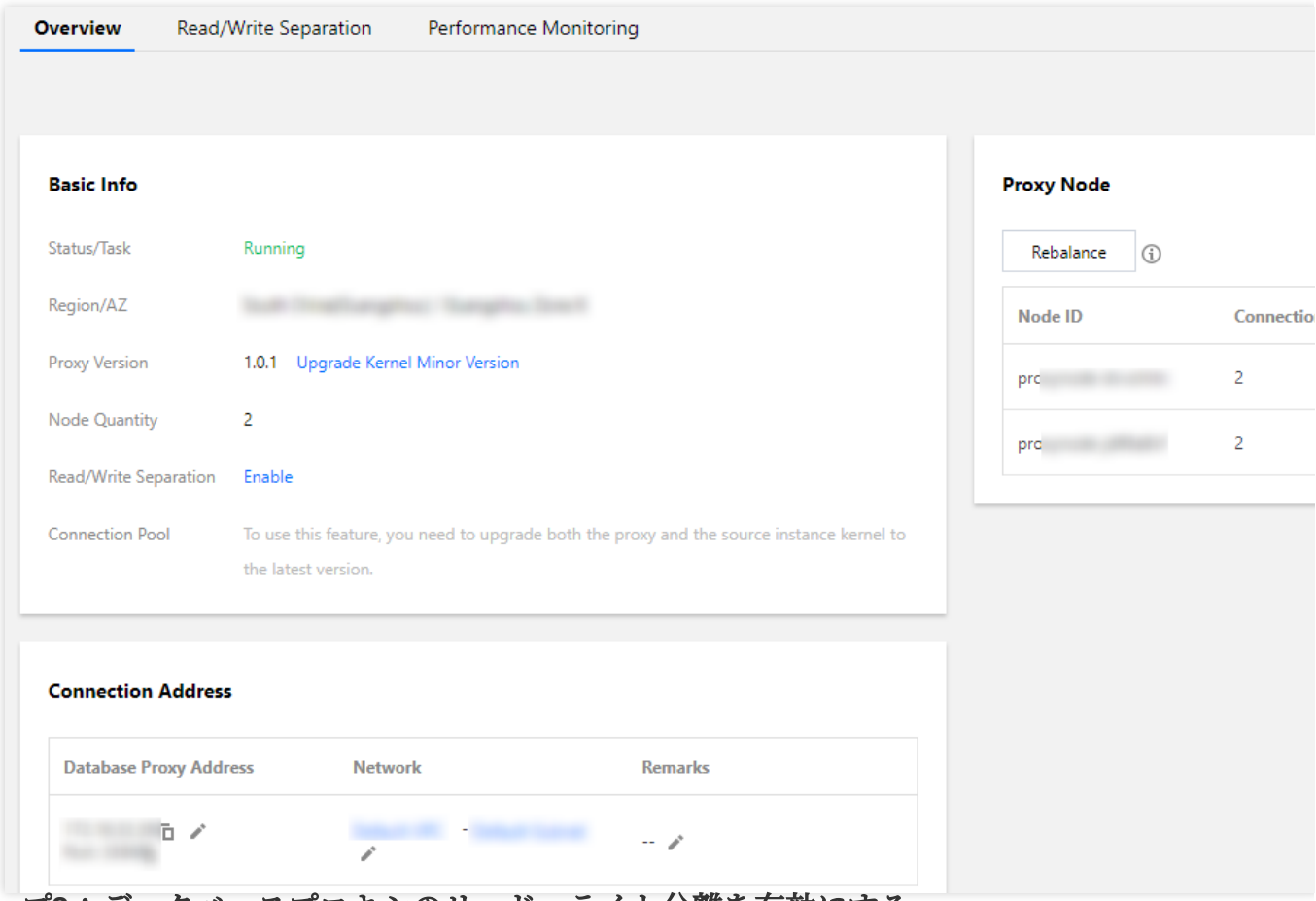
3. ポップアップしたダイアログボックスで、仕様ノードを選択し、**OK**をクリックし、ページを更新します。
ネットワークタイプ：VPCのみをサポートします。デフォルトではマスターインスタンスと一致します。
プロキシ仕様：2コア4000MBメモリ、4コア8000MBメモリ、8コア16000MBメモリ。
ノードの個数：プロキシノードの個数。推奨プロキシ個数は、マスターインスタンスと読み取り専用インスタンスのCPUコア数合計の1/8（切り上げ）です。例えば、マスターインスタンスが4コアのCPUで、読み取り専用インスタンスが8コアのCPUの場合、推奨プロキシ数＝(4+8)/8 ≒2となります。
接続プール状態：接続プールは短い接続が頻繁に新しい接続を確立することで、インスタンスの負荷が高くなるという問題を効果的に解決できます。
セキュリティグループ：重要なネットワークセキュリティ隔離手段であり、必要に応じて既存のセキュリティグループまたはセキュリティグループの新規作成を選択することができます。



4. 有効になると、データベースプロキシページで基本情報の表示、プロキシノードの管理、データベースプロキシアドレスの変更、構成の調整などを行うことができます。

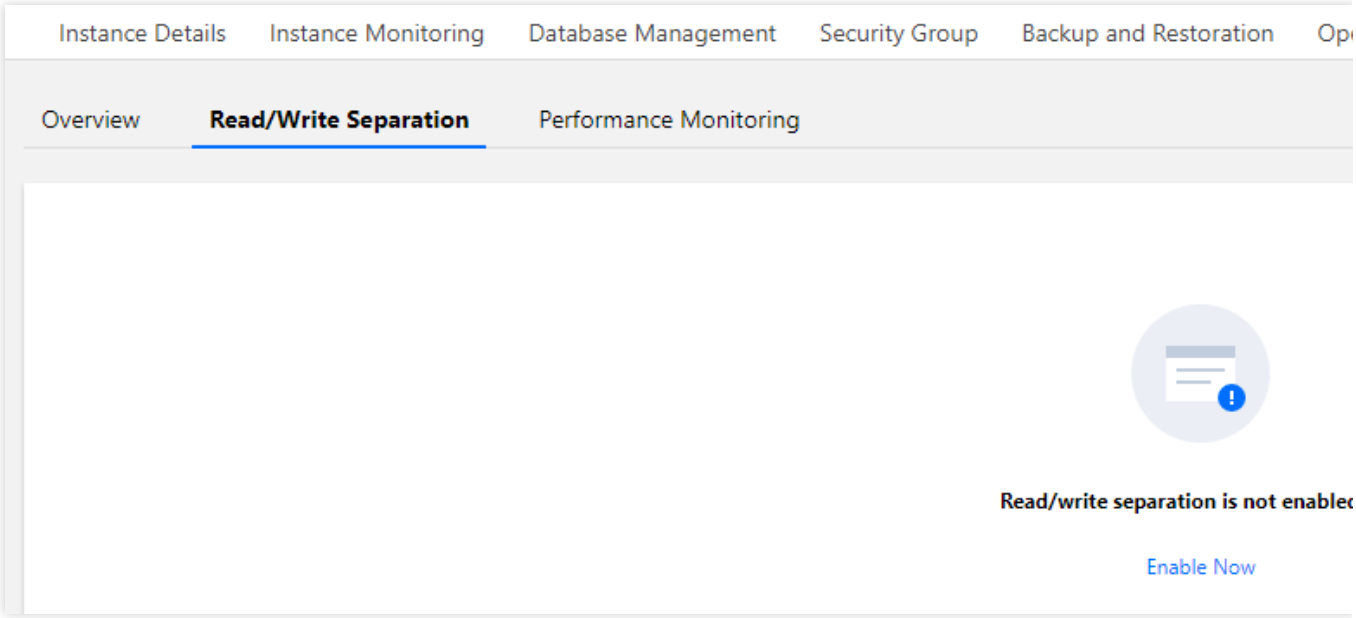
説明：

プロキシノードリストの**接続数**を確認するか、各プロキシノードのパフォーマンス監視を確認することで、各ノードへのアクセスが偏っているかどうかを判断することができます。各プロキシノードへの接続数が偏っている場合、**リロードバランシング**をクリックして接続を分散させることができます。
リロードバランシングはプロキシノードの再起動をトリガーし、再起動中は一時的にサービスが利用できなくなります。 オフピーク時にサービスを再起動することをお勧めします。業務に再接続メカニズムが備わっていることを確認してください。



ステップ2：データベースプロキシのリード・ライト分離を有効にする

1. MySQL コンソールにログインします。
2. 上側でリージョンを選択し、インスタンスIDまたは**操作**列の**管理**をクリックすると、インスタンス管理画面が表示されます。
3. インスタンス管理画面の**データベースプロキシ**で、**読み取り/書き込み分離**を選択し、**今すぐ有効にする**をクリックします。



4. 表示されたダイアログボックスで、設定を選択して、**OK**をクリックします。

ご注意：

状態が実行中のマスターインスタンスと読み取り専用インスタンスのみ、データベースプロキシに追加することができます。

現在、クロスリージョンROと遅延ROは、データベースプロキシの下にマウントすることはできません。

読み取り専用インスタンスの排除：排除ポリシーを有効にするかを設定します。読み取り専用インスタンスでレプリケーションの異常（レプリケーション遅延、レプリケーション中断）が発生した時、データベースプロキシは読み取り専用インスタンスを読み取り/書き込み分離から一時的に外します。遅延排除閾値はデフォルトで10秒で、読み取り専用インスタンスの最小保留数は1個となります。

説明：

読み取り専用インスタンス排除は閾値とインスタンスの最小保留数を設定後、新たな接続に対してのみ有効となります。

遅延排除閾値：読み取り専用インスタンスがマスターインスタンスのデータと同期を取るときに許容される最大遅延時間。読み取り専用インスタンスの遅延時間が閾値を超えた場合、この読み取り専用インスタンスのウェイトがいくらかにもかかわらず、読み取りリクエストはこの読み取り専用インスタンスに転送されません。値の範囲は1以上の整数です。

読み取り専用インスタンスの遅延が閾値を超えた時は排除され、排除されたインスタンスの重みは自動的に0に設定されます。また、システムがユーザーにアラームを出します（まず「データベースプロキシマウントノードの排除」アラームをサブスクリプトしてください。設定については アラーム機能をご参照ください）。

読み取り専用インスタンスの遅延が閾値より小さい時は、新たにデータベースプロキシに追加されます。同時に、遅延排除機能が有効かどうかに関わらず、読み取り専用インスタンスの障害が排除された後、インスタンスが修復されるのを待って新たにデータベースプロキシに追加されます。

読み取り専用の最小保留数の設定により、データベースプロキシが現在のルート中の読み取り専用インスタンスが設定された値よりも小さいことがわかった時は、読み取り/書き込み分離に関与する読み取り専用インスタンス

の個数が最小保留数を満たすまで、異常のある読み取り専用インスタンスを読み取り/書き込み分離に戻します。

ご注意：

読み取り専用データベースで致命的な障害（クラッシュなど）が発生した時、最小保留数を致命的な障害が発生したインスタンスに適用できません。

読み取り専用インスタンス最小保留数：確保しなければならないインスタンスの最小数。既存の読み取り専用インスタンス数がこの最小値以下であり、かつ遅延時間が閾値を超えた場合、既存の読み取り専用インスタンスはいずれも排除されません。

読み取りウェイト割当：インスタンスに読み取りウェイトを割り当てます。システムによる自動割当かカスタムを指定できます。割り当てるウェイトの範囲は0 - 100の整数です。読み取りウェイト割当を設定すると、直ちにすべての接続に反映されます。

データベースプロキシは、重みの設定に従って読み取りリクエストのトラフィックを割り当てます。例えば、2つの読み取り専用データベースの重みがそれぞれ10と20の場合、それらの読み取りリクエストのトラフィックは1：2の比率で割り当てられます。

重みは読み取りリクエストの重みのみで、書き込みリクエストはマスターデータベースに直接ルーティングされ、重みの計算には関与しません。 例えば、クライアントが10個の書き込みステートメントと10個の読み取りステートメントを送信し、マスターデータベースと読み取り専用データベースの重み比が1:1の場合、マスターデータベースは10個の書き込みステートメントと5個の読み取りステートメントを受信し、読み取り専用データベースは5個の読み取りステートメントのみ受信することになります。

システムによる重みの自動割り当てを選択した時、システムはインスタンスのCPUとメモリの仕様に基づいて自動的に重みを割り当てます。このときに設定できるのはマスターインスタンスの重みのみです。

読み取り専用インスタンスの重みが0の場合、データベースプロキシはこの読み取り専用インスタンスへの接続を構築することはありません。読み取り専用インスタンスの重みを0から非0に変更しても、重みはすぐに有効とならず、新規接続に対してのみ有効となります。

フェイルオーバー：有効にするかを設定します。有効にすることを推奨します。読み取り専用インスタンスに異常が発生した場合、データベースプロキシは読み取リクエストをマスターインスタンスへ送信します。

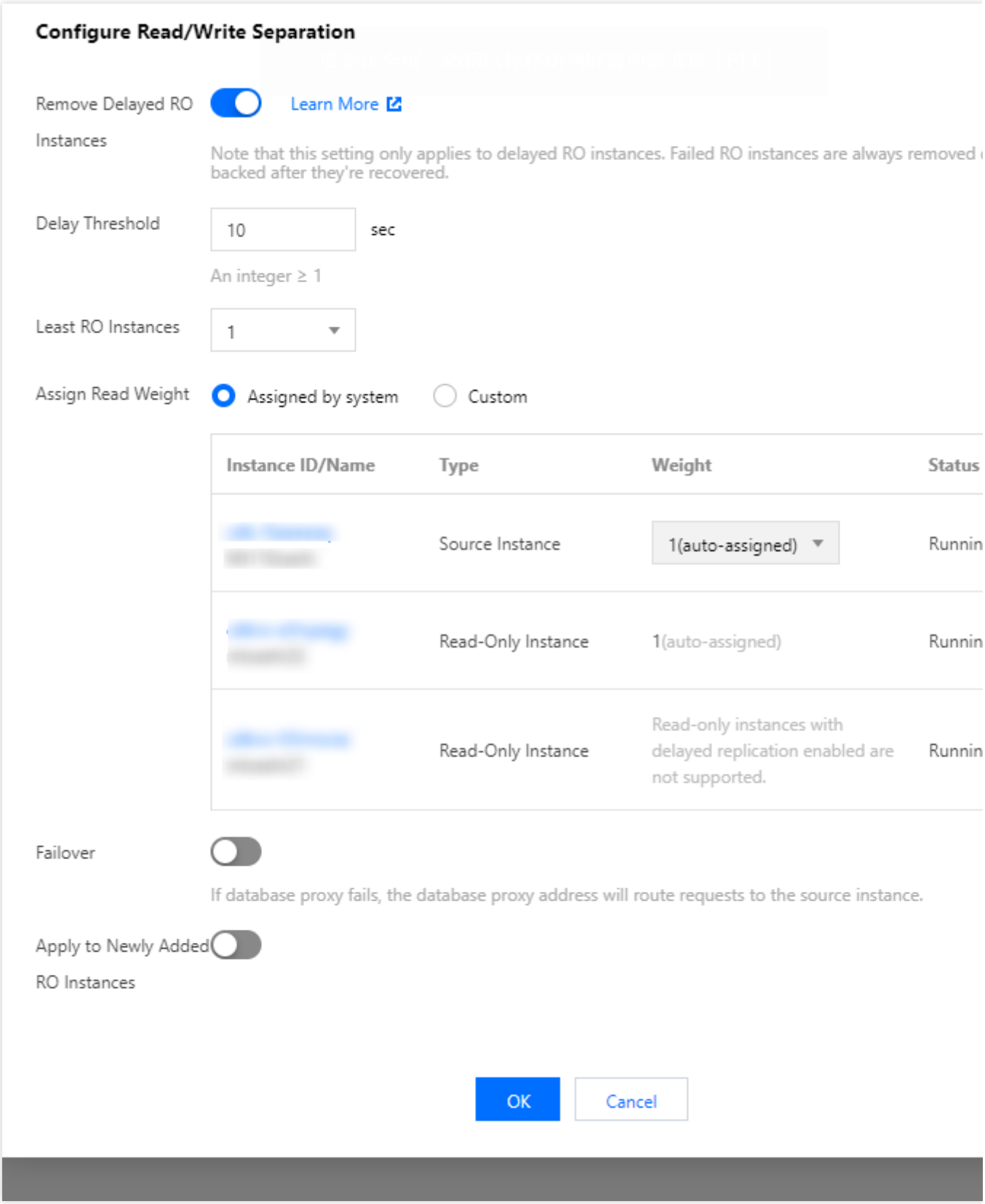
説明：

フェイルオーバー設定後は、新規接続に対してのみ有効となります。

読み取り専用インスタンスの自動追加：有効にするかを設定します。有効に設定した場合、新しい読み取り専用インスタンスを購入すると、自動的にデータベースプロキシに追加されます。

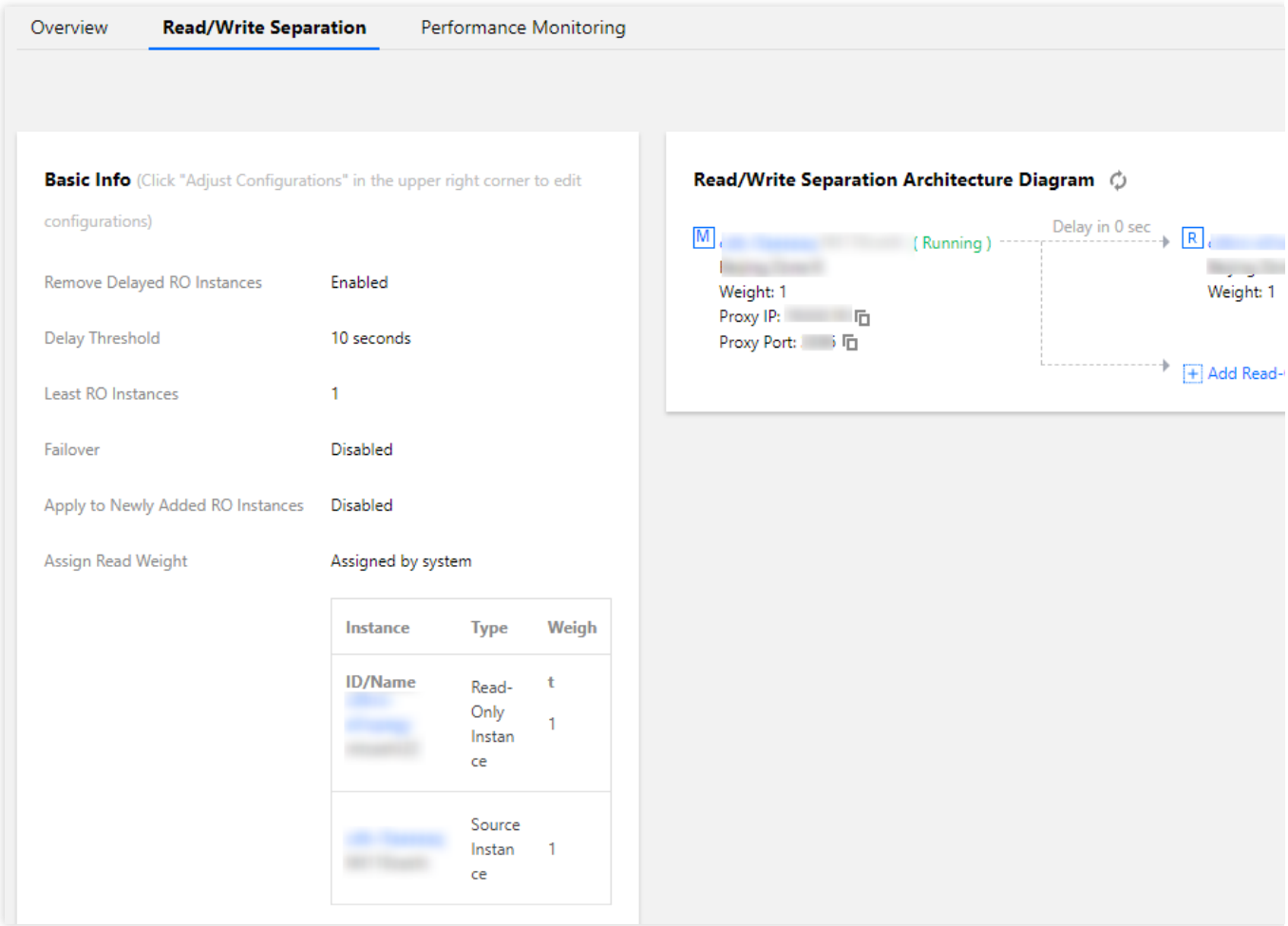
読み取り重みがシステムによる自動割当の場合、新たに購入した読み取りインスタンスに対して、仕様に応じたデフォルト重みを割り当てます。

読み取り重みがカスタムの場合、新たに購入した読み取りインスタンスを追加する時、その重みはデフォルトで0とします。データベースプロキシ読み取り/書き込み分離を調整することで、重みを変更できます。



データベースプロキシのリード・ライト分離を有効にすることに成功した

データベースプロキシのリード・ライト分離が正しく有効になると、データベースプロキシページは次のとおりです。



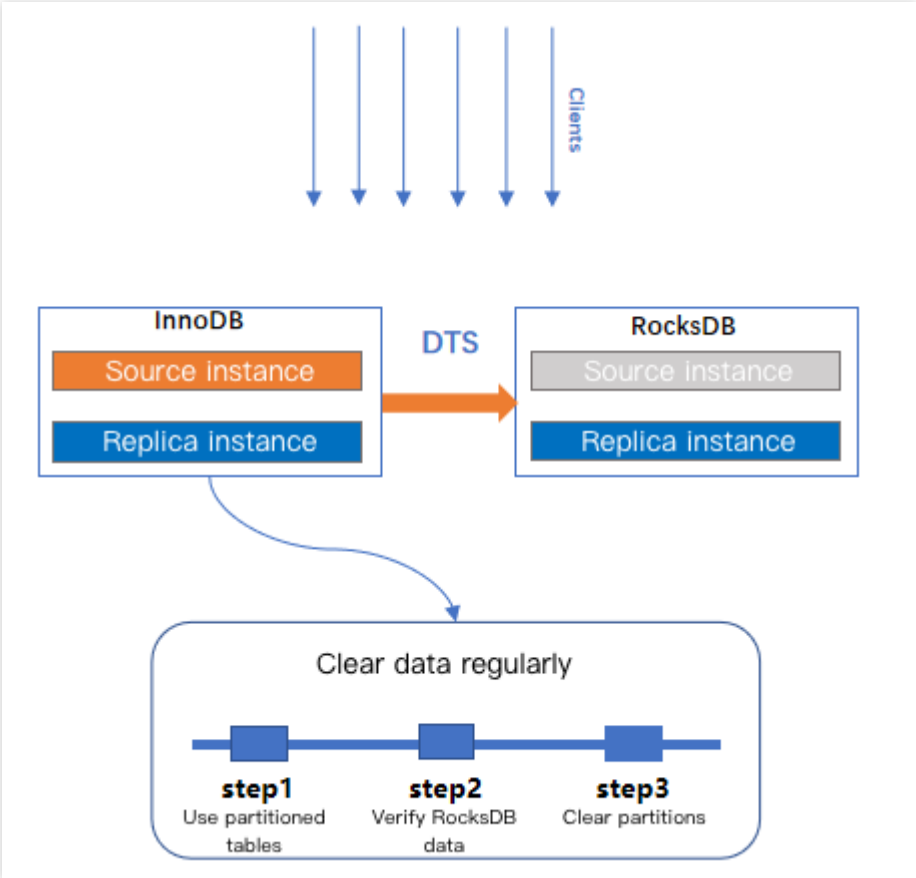
関連ドキュメント

データベースプロキシ接続アドレスの設定
データベースプロキシネットワークの切り替え
セッションレベルの接続プールの設定

DTSでInnoDBデータをRocksDBに移行します

最終更新日：：2024-07-25 17:56:18

RocksDBは非常に人気の高い高性能の永続的なKV (key-value)ストレージであり、TXRocksはTencent TXSQLチームがこれに基づいて開発したトランザクション型ストレージエンジンです。
InnoDBが用いているB+Treeインデックス構造に比べて、TXRocksに使用されるLSM Treeインデックス構造はかなりの割合のメモリスペースを節約できます。InnoDBのB+Treeのスプリットは通常、ページが半分格納となり、ページ内の空き領域が無駄になり、ページ有効利用率が低くなります。RocksDBのSSTファイルは一般的にMB単位以上の大きさに設定されており、ファイルの4Kアライメントの場合、発生する無駄の割合が低く、SST内部もBlockに分割されていますが、Blockはアラインする必要はありません。また、RocksDBのSSTファイルはプレフィックス圧縮を採用しており、同じプレフィックスは1部しか記録されません。同時にRocksDBの異なるレイヤーのSSTは異なる圧縮アルゴリズムを採用することができ、さらにストレージスペースのオーバーヘッドを下げることができます。通常、ストレージ容量を50%削減できます。
データ転送サービスDTSを使用してInnoDBデータをRocksDBに同期することで、書き込みパフォーマンスの向上とストレージ容量の節約を実現できます。



説明：

データ転送サービス(Data Transmission Service、DTS)は、データ移行、データ同期、データサブスクリプションを一体的に提供するデータベースデータ転送サービスです。業務を中断することなくデータベースを簡単に移行でき、リアルタイムの同期チャネルを使用して、リモートディザスタリカバリのための高可用性データベースアーキテクチャを簡単に構築でき、データサブスクリプションによって提供されるCDBを使用して、データをリアルタイムで増分更新することができます。また、増分データはビジネスニーズに合わせて自由に消費することができます。

注意事項

データ転送サービスDTSを使用してデータ同期を行い、増分データを確実にRocksDBに同期させ、Delete操作をブロックします。
RocksDBにデータを転送した後、データの有効性を確認し、有効性を確認した後にソースライブラリからデータをクリーンアップして、ストレージ使用量を削減します。
ソーステーブルとしてパーティションテーブルを使用すれば、データ・クリーンアップの効率を向上させることができます（パーティションテーブルの使用制限に注意してください。ターゲットテーブルではパーティション化がサポートされていません）。
RocksDBが定期的にcompactionを実行することで、データ領域の消費を効果的に削減できます。
RocksDBエンジンの注意事項については、TXRocksエンジンのご使用にあたっての注意事項をご参照ください。

操作手順

1. DTSコンソールにログインし、左側のナビゲーションで**データ移行**ページを選択し、**移行タスクの新規作成**をクリックして移行タスク新規作成ページに移動します。
2. 「移行タスクの新規作成」ページで、移行のソースインスタンスのタイプと所属リージョン、ターゲットインスタンスのタイプと所属リージョンや仕様などを選択し、**今すぐ購入**をクリックします。

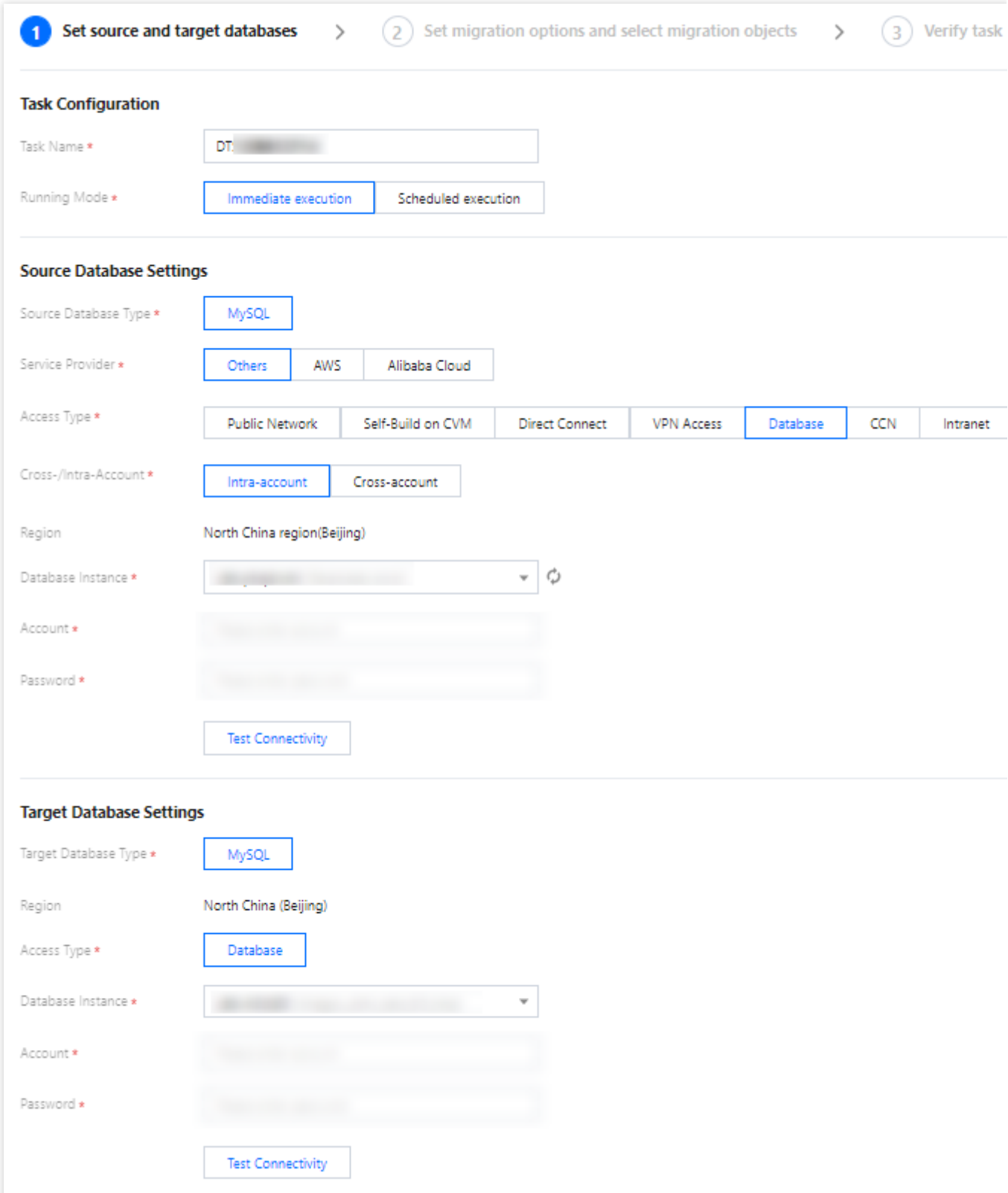
設定項目	の説明
ソースインスタンスのタイプ	ソースデータベースの種類に応じて選択してください。購入後に変更することはできません。ここで「MySQL」を選択します。
ソースインスタンスのリージョン	ソースデータベースの所属リージョンを選択します。ソースライブラリがカスタムデータベースの場合は、カスタムデータベースに最も近いリージョンを選択します。
ターゲットインスタンスのタイプ	ターゲットデータベースのタイプに応じて選択してください。購入後に変更することはできません。ここで「MySQL」を選択します。
ターゲットインスタンスのリージョン	ターゲットデータベースの所属リージョンを選択します。

仕様	ビジネスの状況に応じて移行リンクの仕様を選択します。仕様別のパフォーマンスと課金の詳細については、課金の概要をご参照ください。

3. 移行タスクの**操作**列で**その他>設定**を選択し、ソースデータベースおよびターゲットデータベースの設定ページで、タスクの設定、ソースライブラリの設定、およびターゲットライブラリの設定を完了し、ソースライブラリとターゲットライブラリの接続がパスしたことをテストしたら、**保存**をクリックします。

説明：

接続テストに失敗した場合は、表示メッセージと修正指導に従ってトラブルシューティングして解決し、再度実行してください。



4. 移行オプションの設定ページおよび移行オブジェクトの選択ページで、移行タイプおよびオブジェクトを設定し、**保存**をクリックします。

説明：

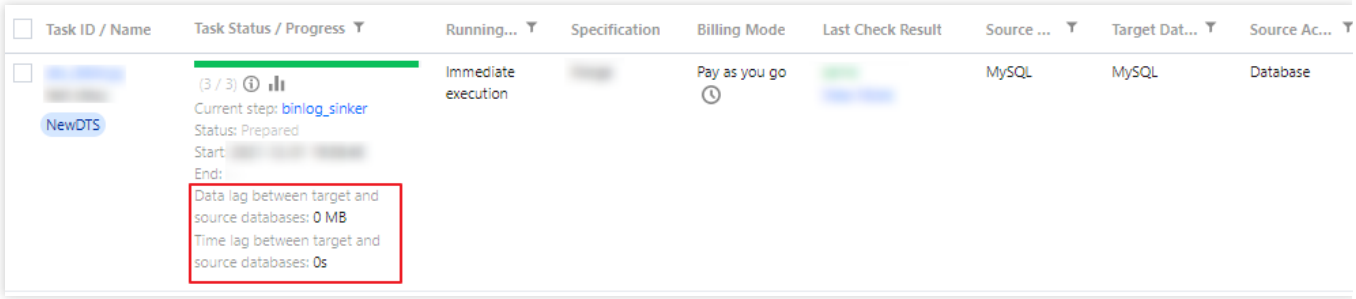
移行中にgh-ost、pt-oscなどのツールを使用してテーブルのOnline DDLを作成するが必要があるとユーザーが判断した場合、**オブジェクトの移行**には、このテーブルだけでなく、テーブルが存在するライブラリ全体（またはイ

ンスタンス全体）を選択してください。そうしないと、Online DDLの変更によって生成されたテンポラリ・テーブルのデータをターゲットデータベースに移行できません。
移行中にテーブルに対してrename操作を使用する必要があるとユーザーが判断した場合（例えば、table Aをtable Bにrenameする）。**オブジェクトの移行**には、table Aだけでなく、table Aが存在するライブラリ全体（またはインスタンス全体）を選択してください。そうしないとエラーが報告されます。

設定項目	の説明
移行のタイプ	お客様のシナリオに合わせて選択してください。 構造移行：データベース内のライブラリやテーブルなどの構造化データを移行します。 フル移行：データベース全体を移行します。移行データは、タスクの開始時にソースデータベースに存在していたもののみを対象とします。タスクの開始後にソースライブラリにリアルタイムで新たに書き込まれたデータは含まれません。 フル移行+増分移行：移行データには、タスクの開始時にソースライブラリに存在していたものが含まれます。また、タスクの開始後にソースライブラリにリアルタイムで新たに書き込まれたデータも含まれます。移行中にソースライブラリにデータが書き込まれ、ダウンタイムなしでスムーズな移行が必要な場合は、このシナリオを選択します。
移行対象	インスタンス全体：インスタンス全体を移行しますが、information_schema、mysql、performance_schema、sysなどのシステム・ライブラリは含まれません。 指定したオブジェクト：指定したオブジェクトを移行します。
指定のオブジェクト	ソースライブラリのオブジェクトから移行するオブジェクトを選択し、選択したオブジェクトボックスに移動します。
アカウントを移行するかどうか	ソースライブラリのアカウント情報を移行する必要がある場合は、この機能をチェックします。

5. 検証タスクページで、検証を実行し、検証タスクが通過したら、**タスクを開始する**をクリックします。
6. データ移行タスクのリストを返します。タスクが実行準備状態になり、1分～2分後にデータ移行タスクが正式に開始されます。
構造移行または**フル移行**を選択します：タスクが完了すると自動的に終了します。手動で終了する必要がありません。
フル移行+増分移行を選択します：フル移行が完了すると、自動的に増分データの同期フェーズに入ります。増分データの同期は自動的に終了しません。手動で**完了**をクリックして増分データの同期を終了させてください。
適切な時間を選択して増分データの同期を手動で完了し、ビジネスの切り替えを完了してください。
移行フェーズが増分同期であることを確認し、無遅延状態を表示し、ソースライブラリの書き込みを数分間停止します。

ターゲットとソース・ライブラリのデータ・ギャップが0MBで、ターゲットとソース・ライブラリの遅延時間が0秒の場合に、増分同期を手動で完了します。



7. 移行タスクのステータスが**タスク成功**に変わるとき、InnoDBデータをRocksDBに同期することができます。

LAMPスタック上のWebアプリケーションの構築

最終更新日：：2024-07-25 17:56:18

LAMP (Linux + Apache + MySQL/MariaDB + Perl/PHP/Python) は、動的サイトまたはサーバーのセットアップによく使用されるオープンソースソフトウェアです。プログラムはすべて独立していますが、一緒に使用されるため、それらの互換性がますます高まり、全体で強力なWebアプリケーションプラットフォームとなっています。
このチュートリアルでは、TencentDBインスタンスを起動し、CVMインスタンスを使用してLAMPアプリケーションを構成して、TencentDBインスタンスの高可用性環境に接続するプロセスについて説明します。
TencentDBインスタンスを実行した後、データベースと環境のライフサイクルを分離します。これにより、複数のサーバーを同じデータベースに接続できるようになり、データベースのメンテナンスを簡略化されるため、データベースのインストール、設定、バージョンの更新および故障対応の問題を気にする必要がありません。
説明：
本チュートリアルで使用したTencentDBインスタンスとCVMインスタンスは同じリージョンにあります。
TencentDBインスタンスとCVMインスタンスが同じリージョンにない場合は、パブリックネットワークアクセスをご参照ください。

TencentDBインスタンスの初期化

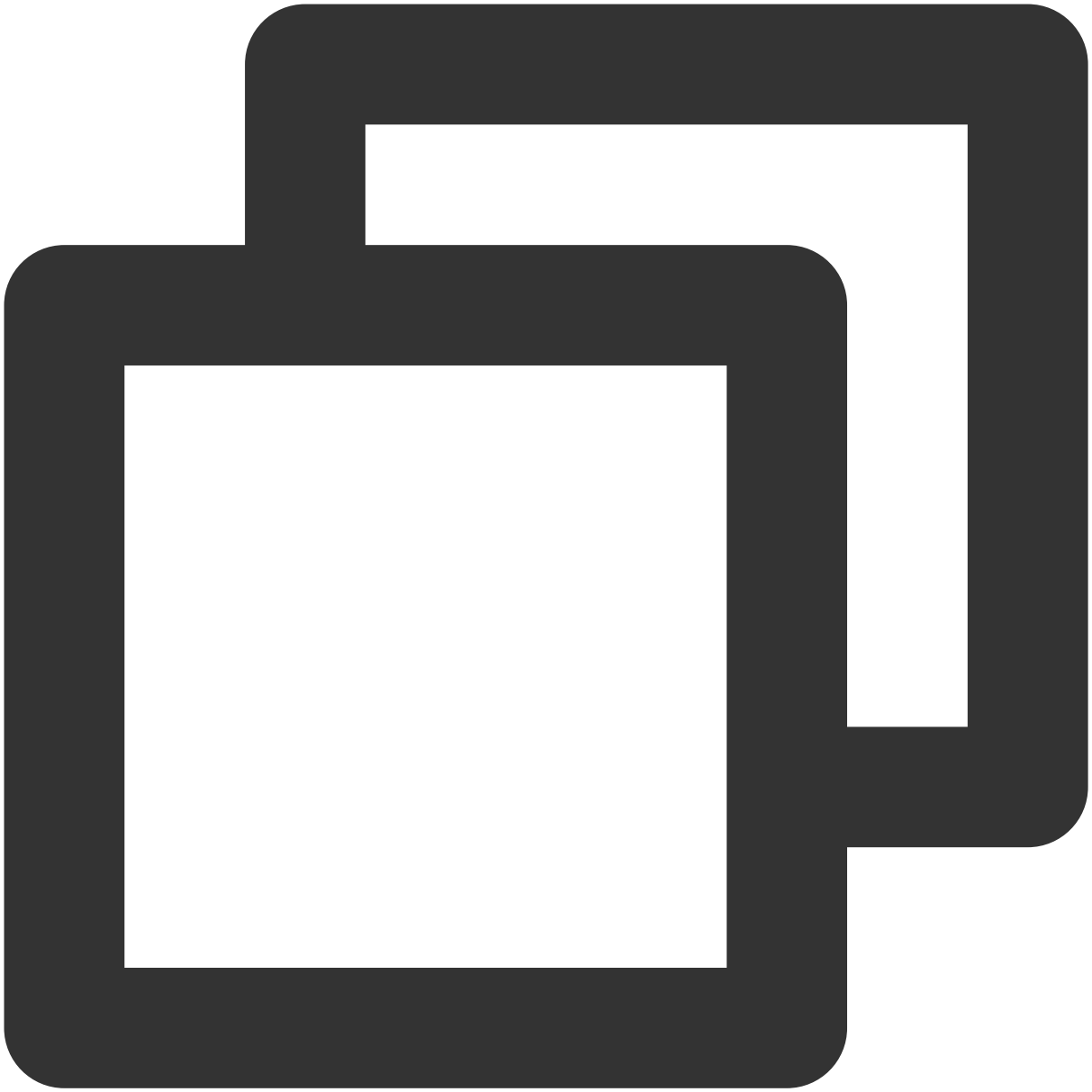
TencentDBインスタンスを購入して初期化する方法の詳細については、購入方式 とMySQLデータベースの初期化をご参照ください。

CVMインスタンスへのログイン

CVMインスタンスの購入とアクセスについては、Linux CVMのクイックスタートをご参照ください。このチュートリアルではCentOSを使用します。

MySQLクライアントのインストール

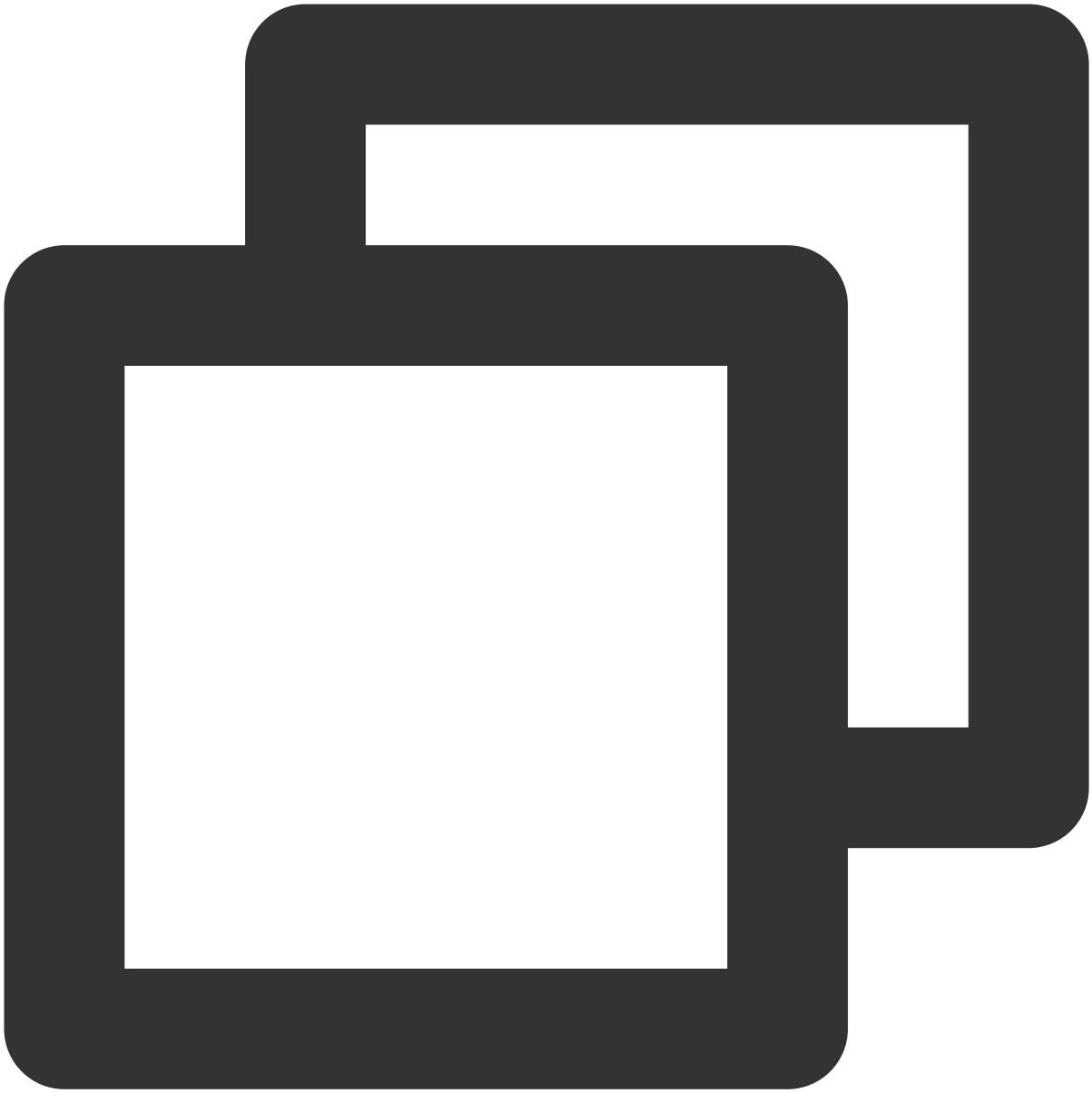
1. CVMインスタンスでは `yum` を使用してMySQLクライアントをインストールします。



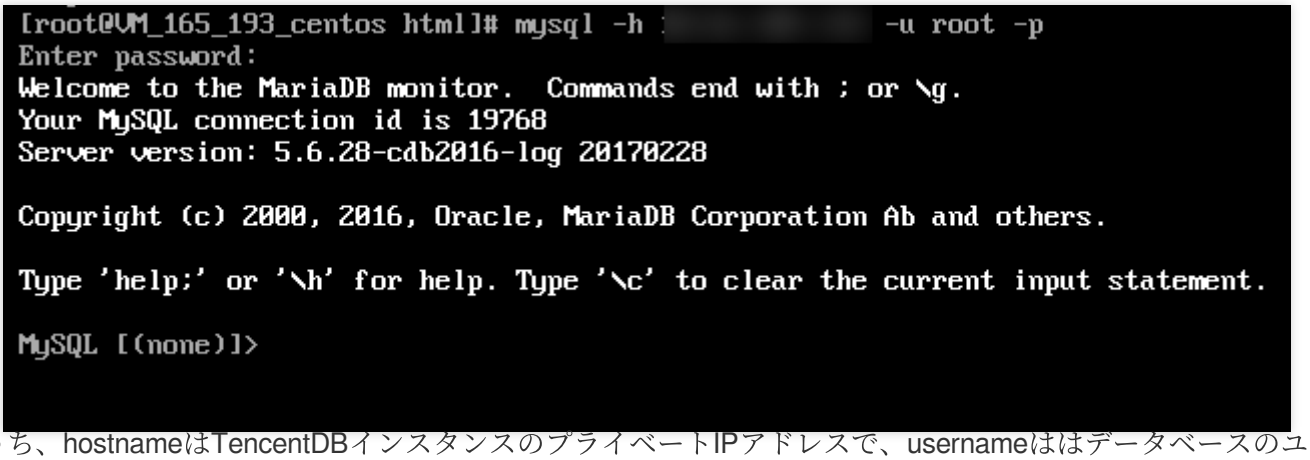
```
yum install mysql -y
```

```
[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
================================================================================
Installing:
 mariadb                    x86_64                     1:5.5.52-1.el7

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. インストール完了後、TencentDBインスタンスに接続します。

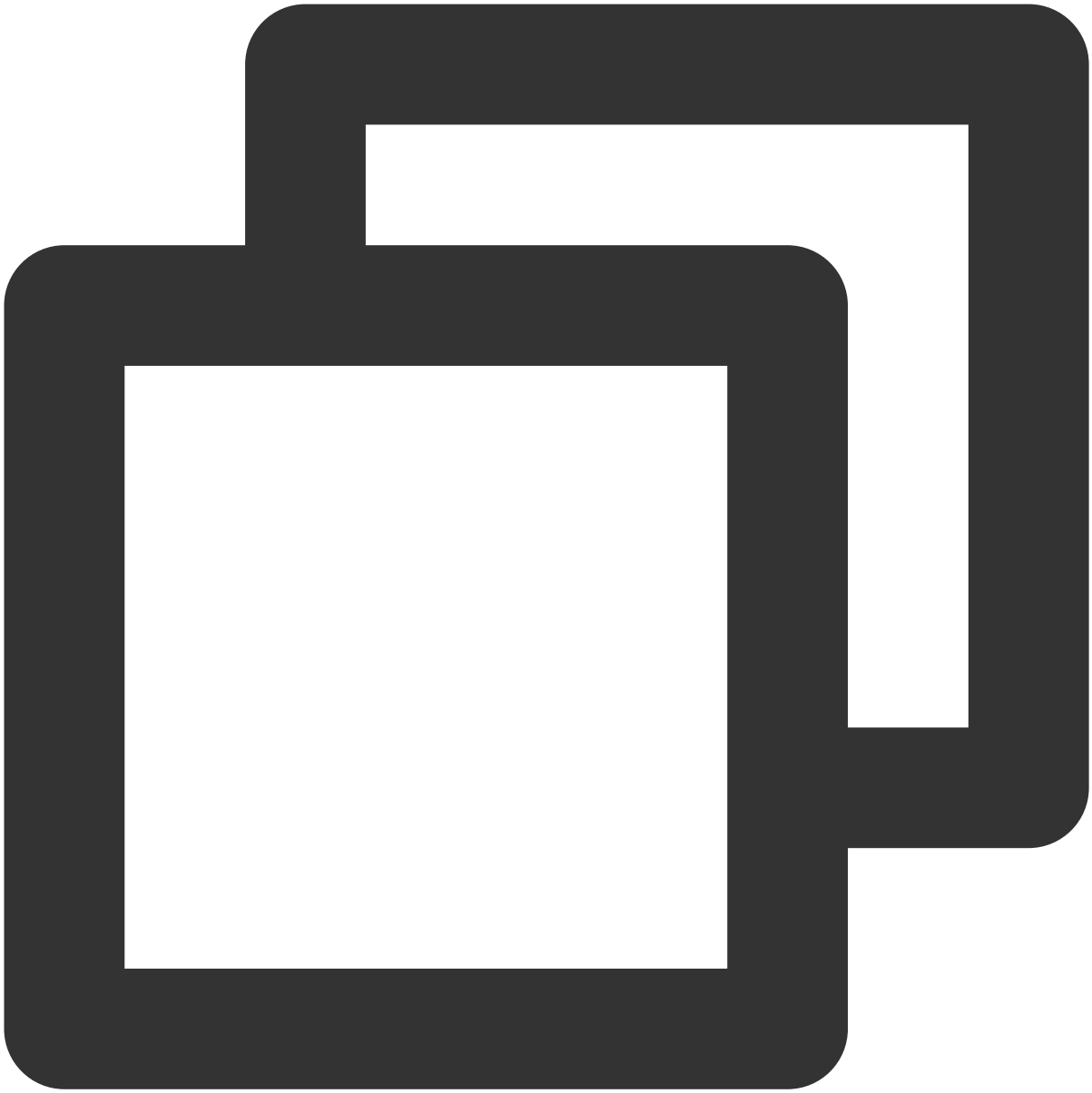


```
mysql -h hostname -u username -p
```



このうち、hostnameはTencentDBインスタンスのプライベートIPアドレスで、usernameははデータベースのユーザー名です。

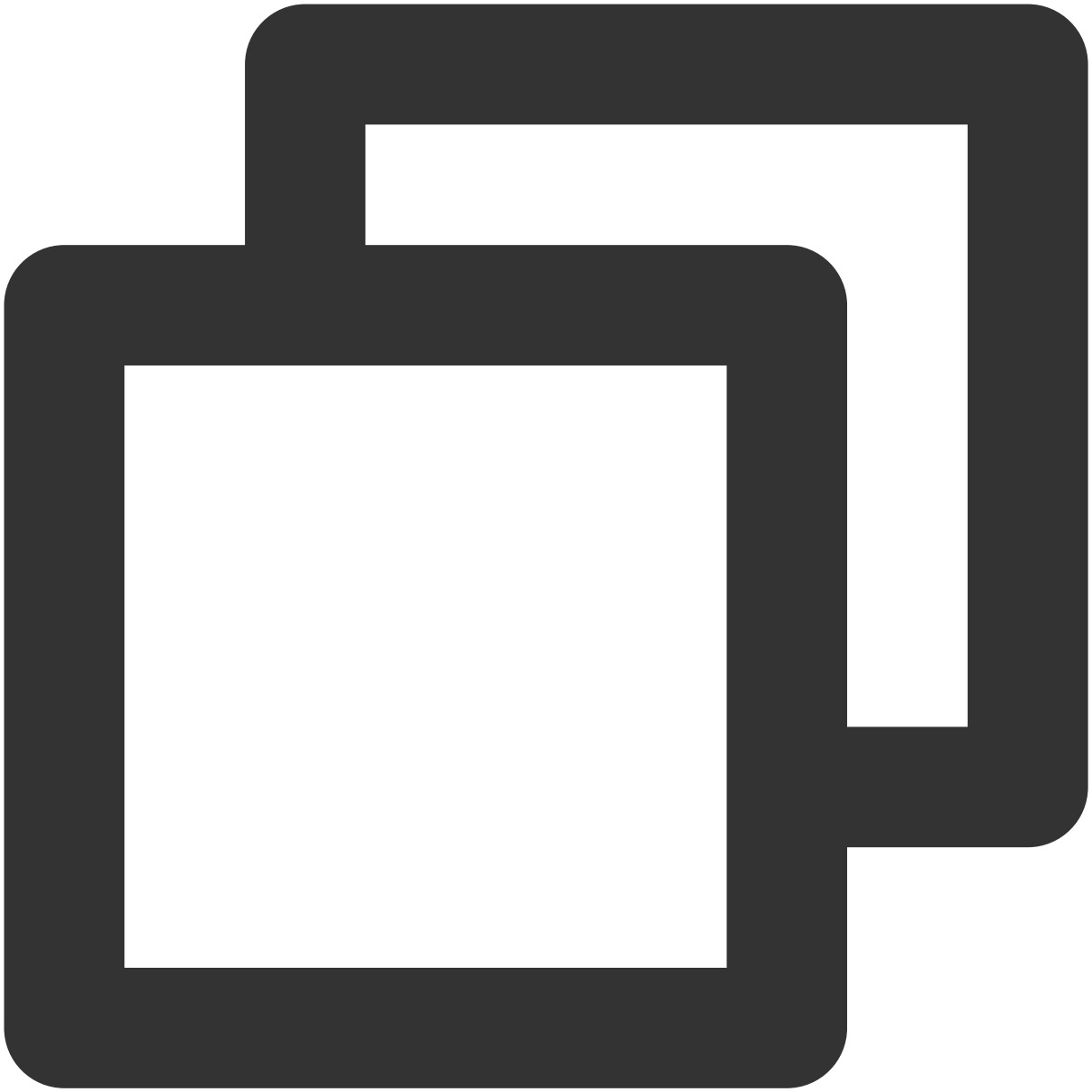
3. 接続が成功したら、インスタンスを閉じて次のステップに進むことができます。



```
quit;
```

Apacheサービスのインストール

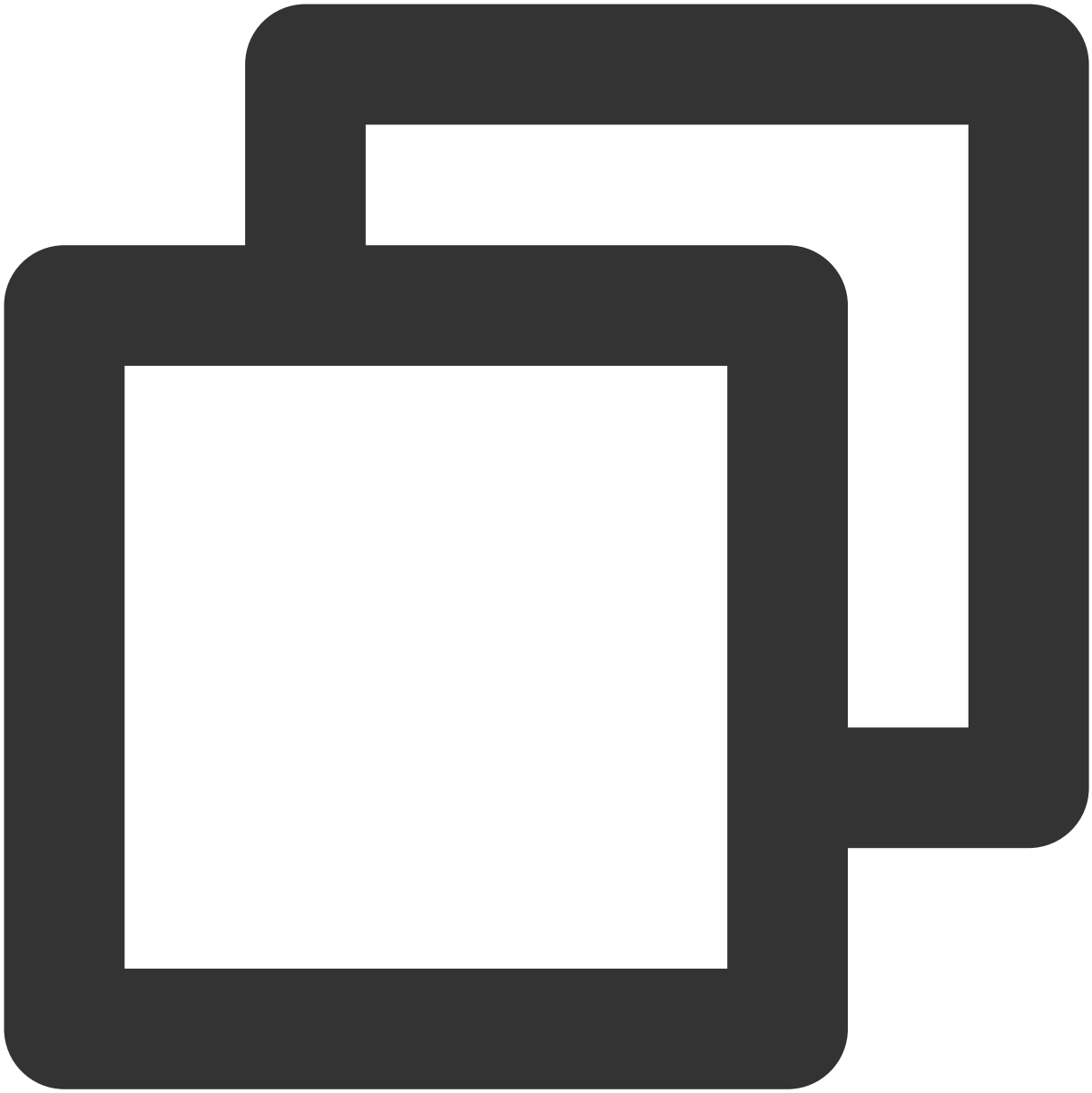
1. CVMインスタンスで `yum` を使用してApacheをインストールします。



```
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. Apacheサービスを起動します。



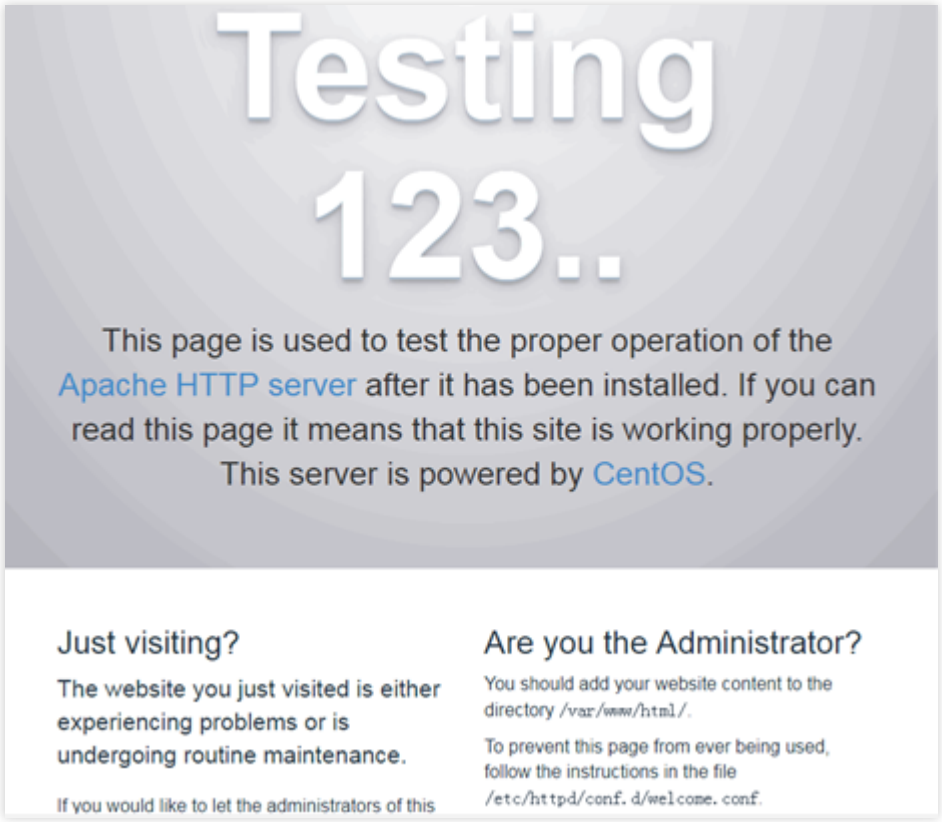
```
service httpd start
```

3. Apacheをテストします。

注意：

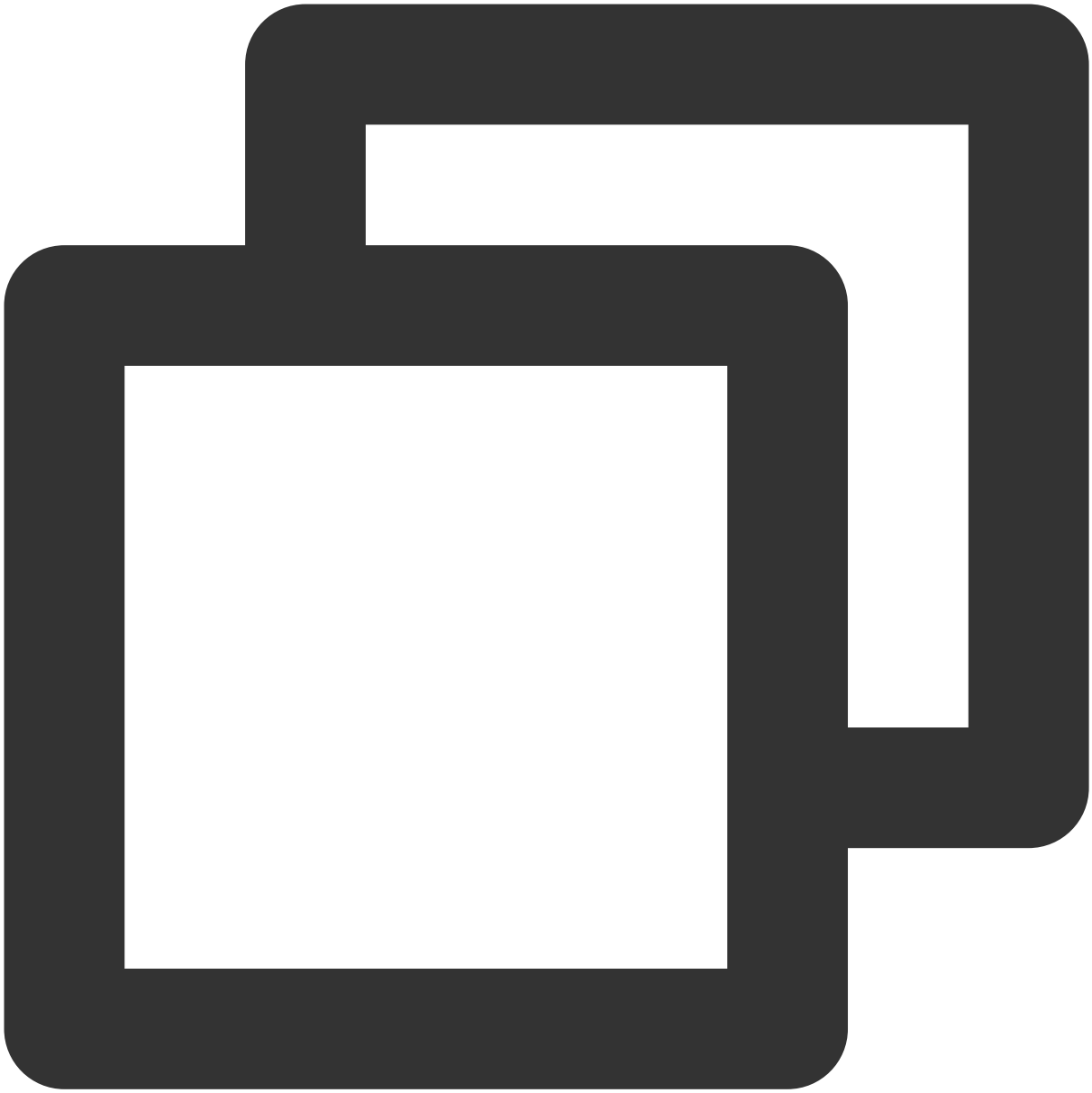
この手順では、CVMインスタンスがセキュリティグループにおいて、ソースが**all**、ポートプロトコルが**TCP:80**であるインバウンドルールを構成する必要があります。セキュリティグループの設定方法の詳細については、セキュリティグループをご参照ください。

ローカルブラウザに `http://115.xxx.xxx.xxx/` （このうち `115.xxx.xxx.xxx` はCVMインスタンスのパブリックIPアドレス）を入力し、次の画面が表示されたら、Apacheの起動に成功しています。



PHPのインストール

1. CVMインスタンスで `yum` を使用してPHPをインストールします。

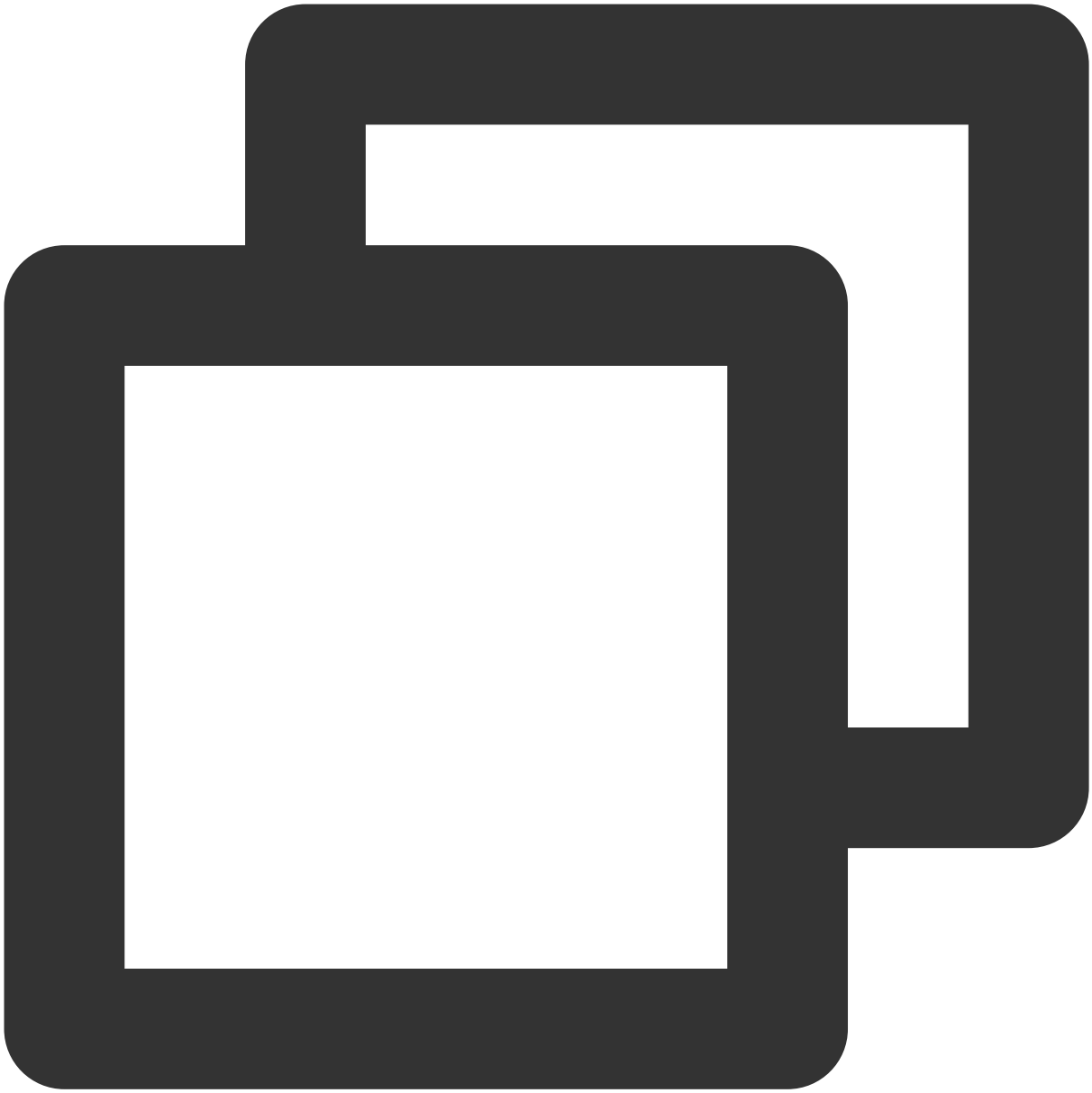


```
yum install php -y
```

```
[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
```

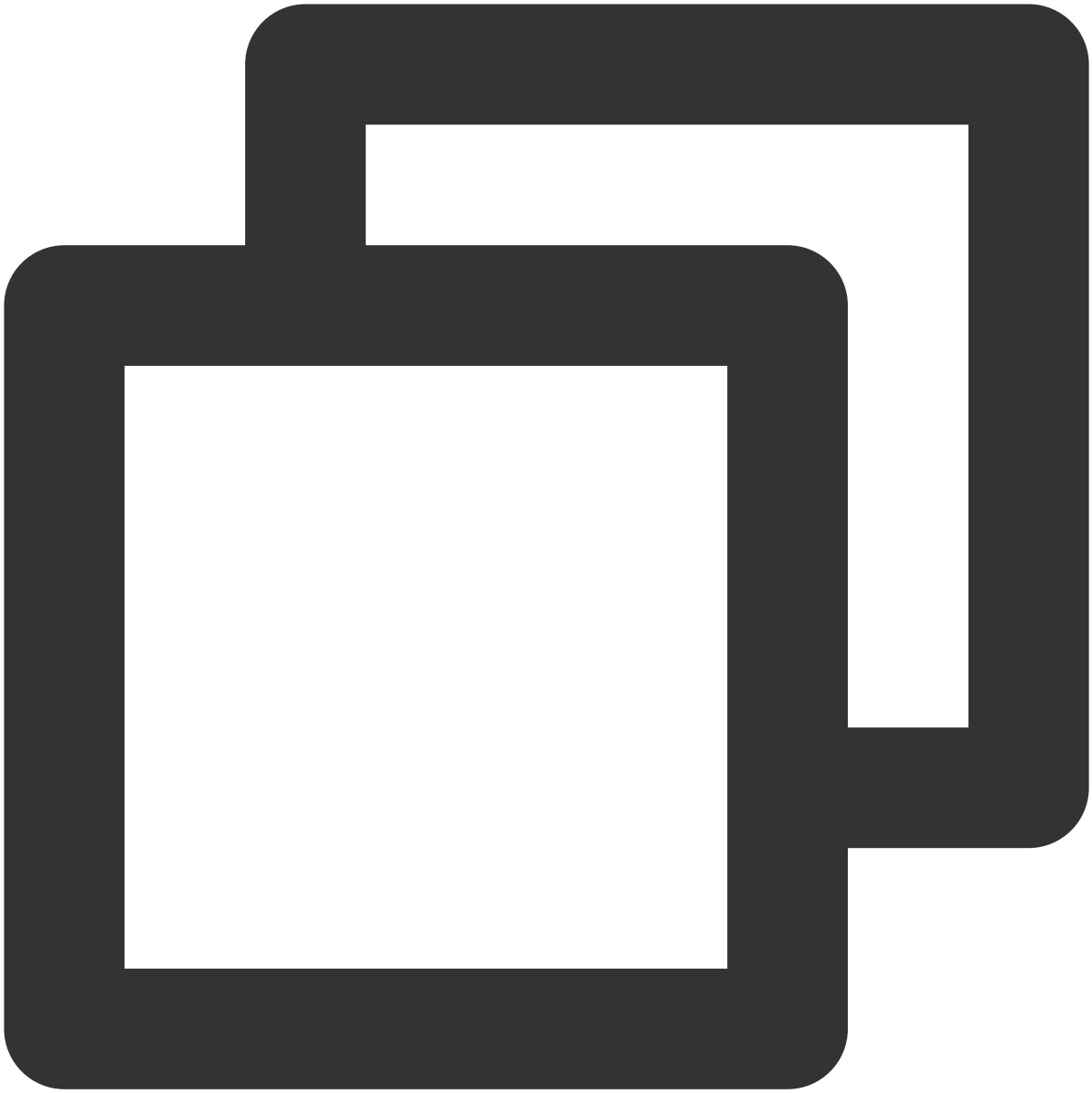
LAMP環境をテストするプロジェクトの作成

1. CVMインスタンスの `/var/www/html` ディレクトリでinfo.phpファイルを1つ作成します。コード例は以下のとおりです。



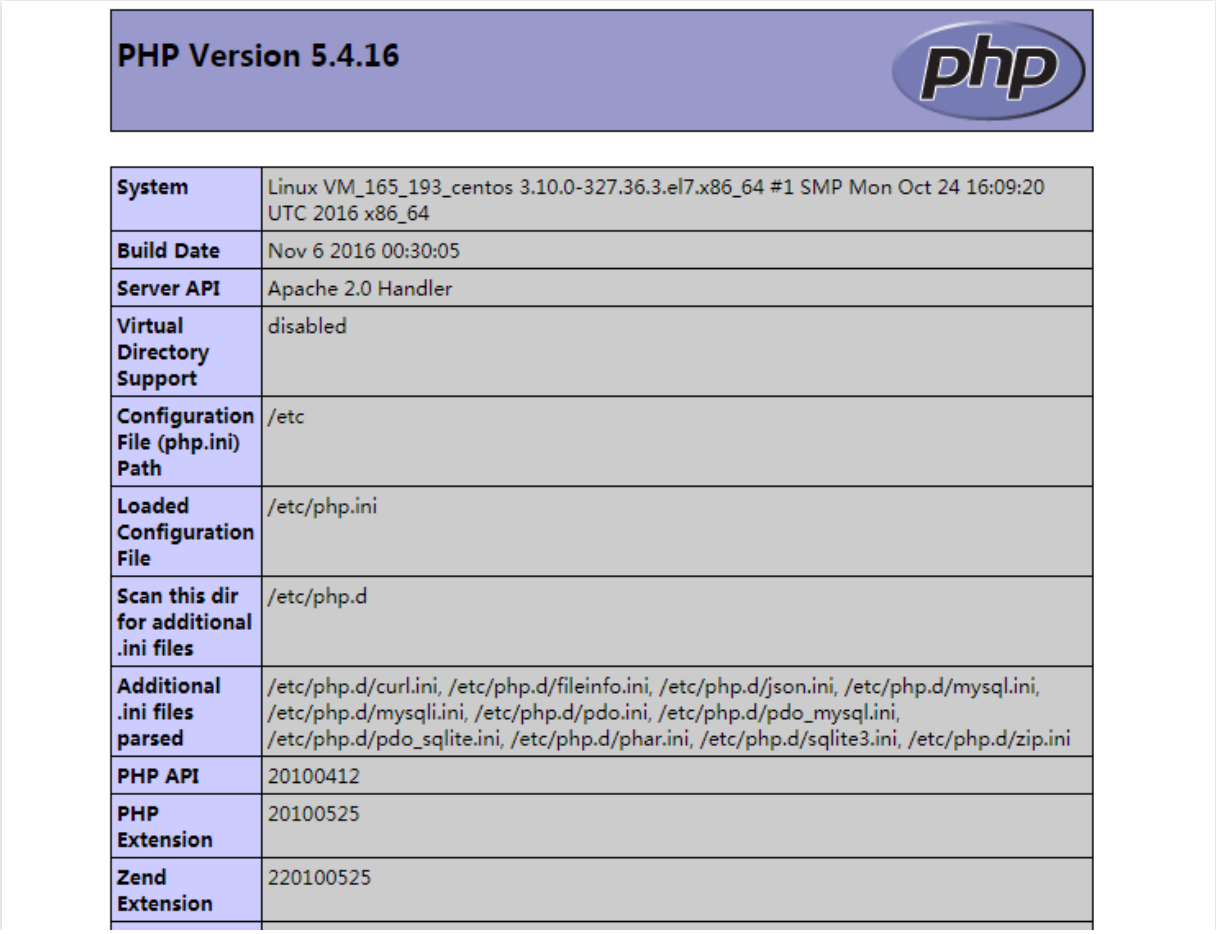
```
<?php phpinfo(); ?>
```

2. Apacheサービスを再起動します。



```
service httpd restart
```

3. ローカルブラウザに `http://0.0.0.0/info.php`（このうち `0.0.0.0` はCVMインスタンスのパブリックIPアドレス）を入力し、次の画面が表示されたら、LAMPサービスのデプロイに成功しています。



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

Drupalウエブサイトの構築

最終更新日：：2024-07-25 17:56:18

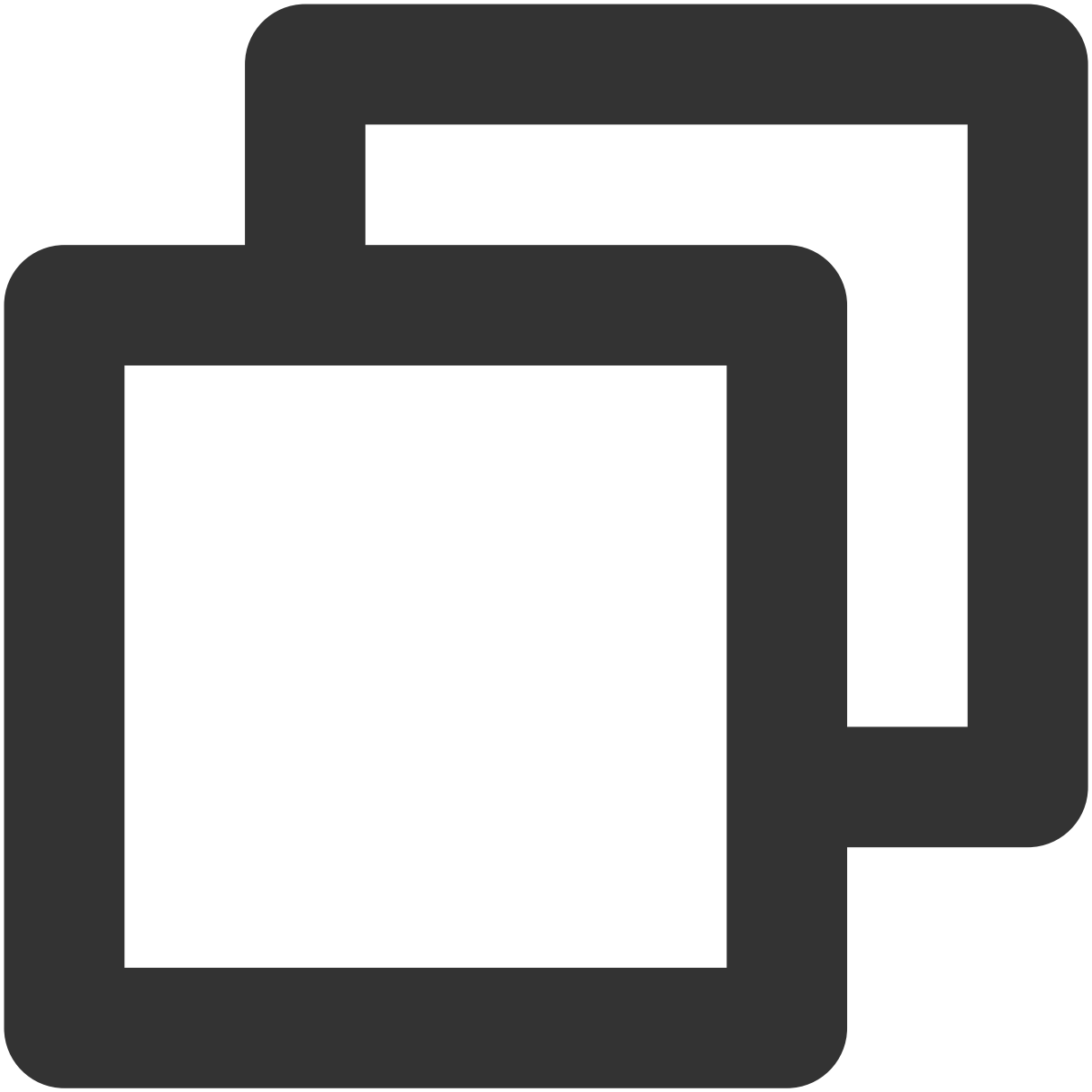
Drupalは、PHP言語で記述されたオープンソースのコンテンツ管理フレームワーク(Content Management Framework)であり、コンテンツ管理システム(Content Management System)とPHP開発フレームワーク(Framework)で構成されます。Drupalは、さまざまな機能とサービスを提供するダイナミックウェブサイトを構築するために使用でき、個人のブログから大規模なコミュニティまで、さまざまなアプリケーションのウェブサイトプロジェクトをサポートできます。
このチュートリアルを通じて、Cloud Virtual Machine(CVM)でDrupal eコマースウェブサイトを構築する方法を学ぶことができます。
使用するソフトウェア環境：CentOS7.2、Drupal7.56、PHP5.4.16。

CVMインスタンスへのログイン

CVMの購入とアクセスについては、Linux CVMのクイックスタートをご参照ください。

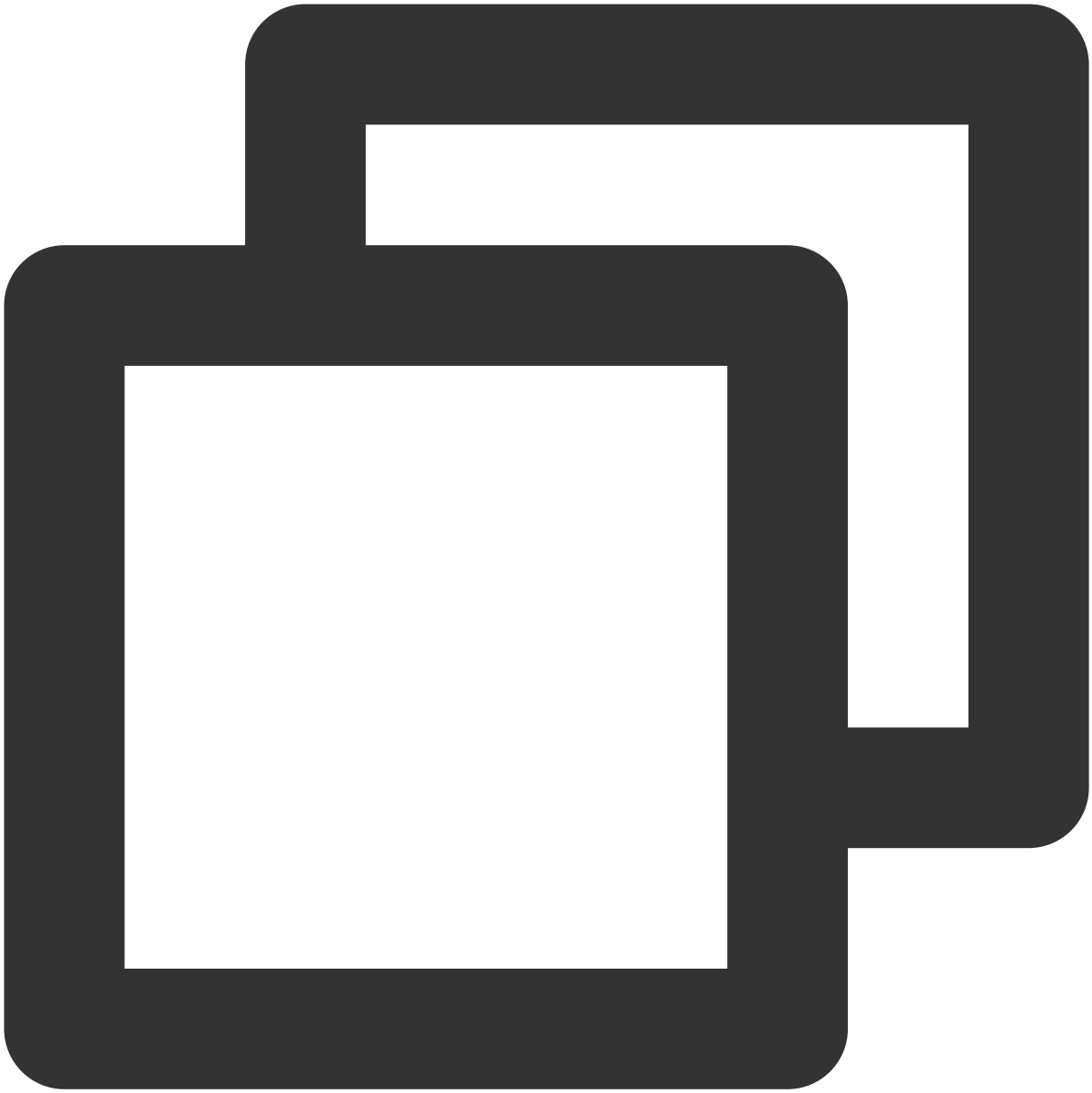
MariaDBサービスのインストール

1. CentOS7以降のバージョンはデフォルトでMariaDBデータベースをサポートするため、MariaDBデータベースを使用します。CVMインスタンスでは、 `yum` を使用してMariaDBサービスをインストールします。



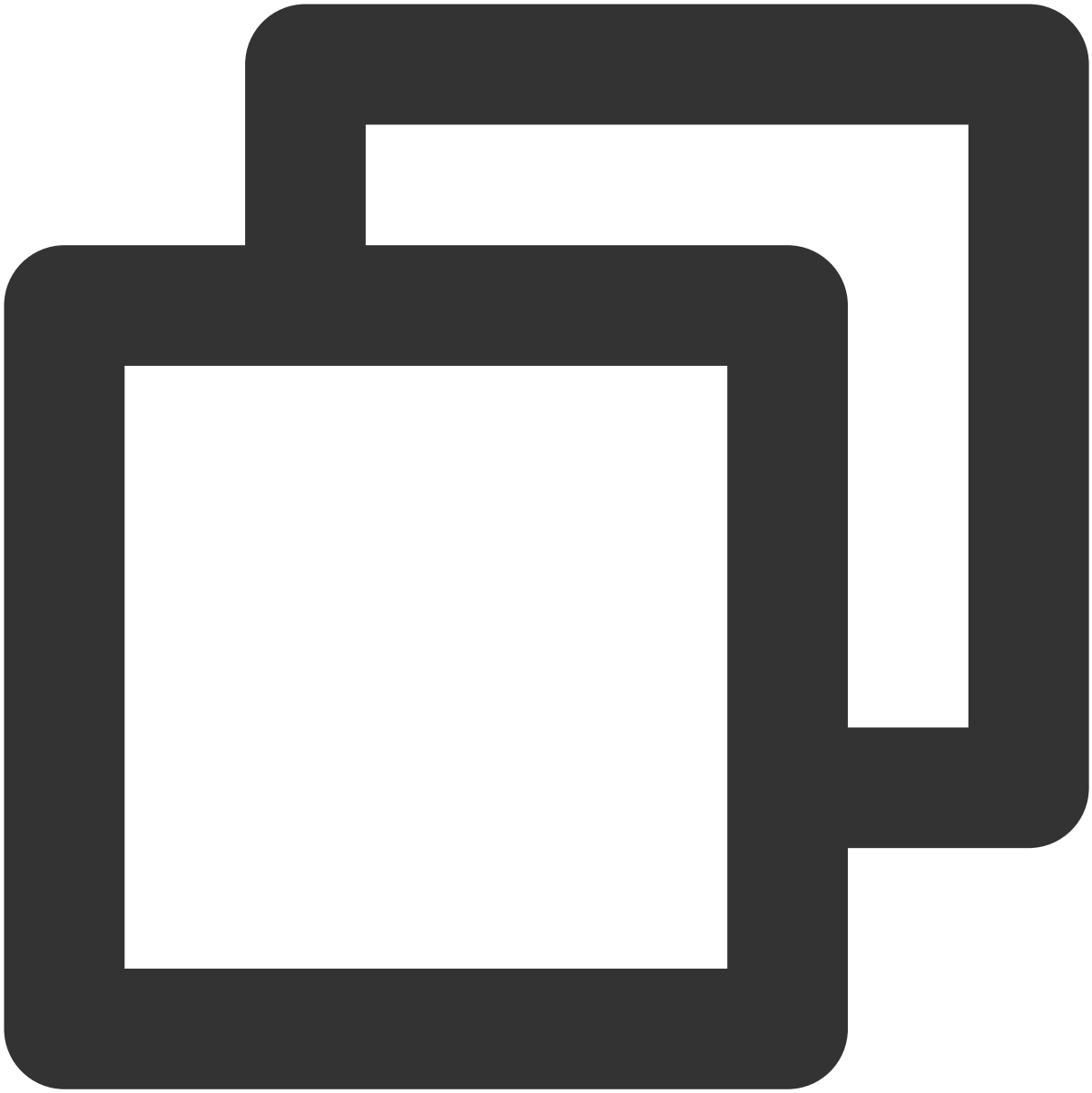
```
yum install mariadb-server mariadb -y
```

2. MariaDBサービスを起動します。



```
systemctl start mariadb
```

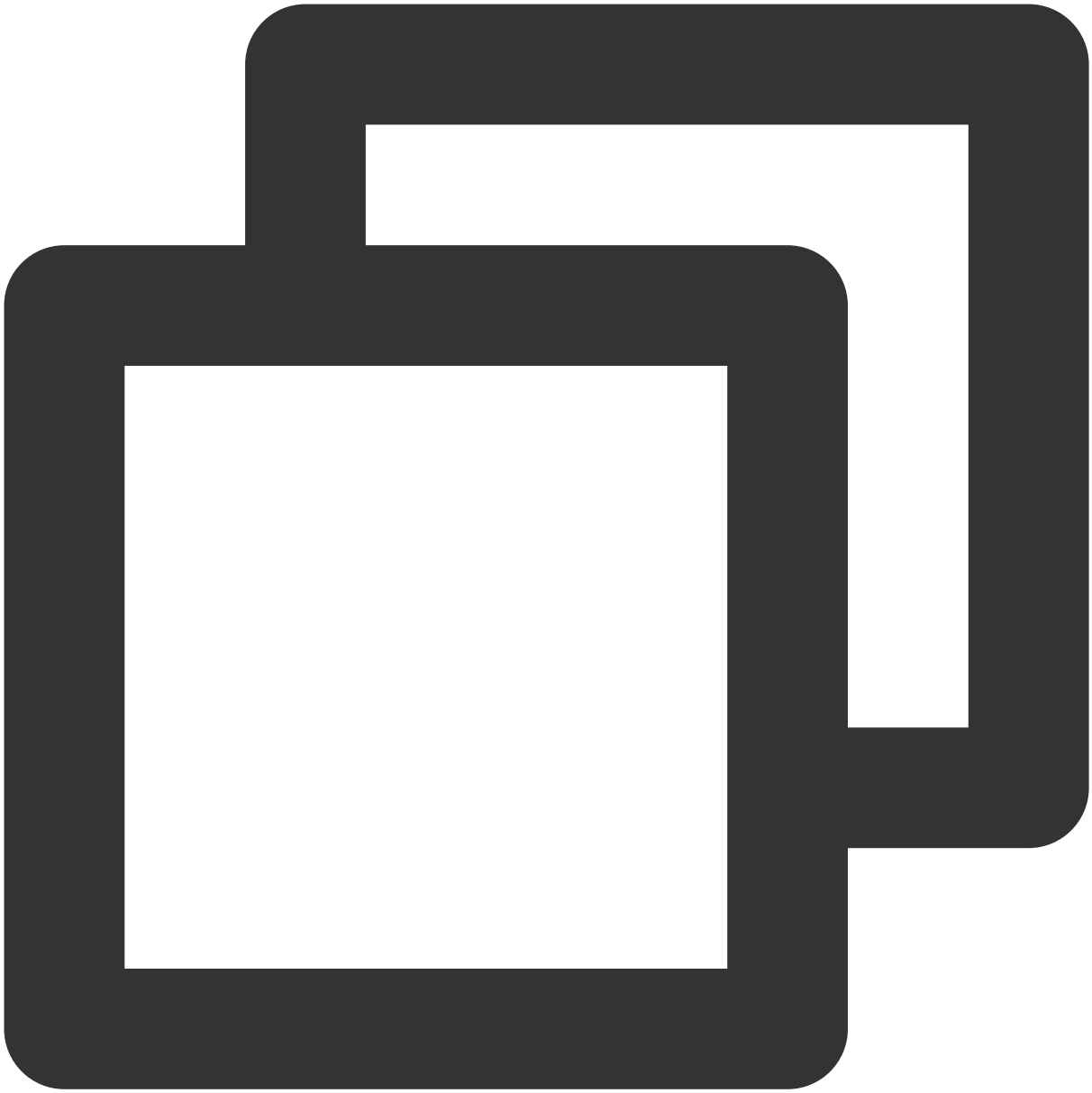
3. Drupal用のデータベースを作成します。（このプロジェクトは、データベース名としてdrupalを使用します）



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

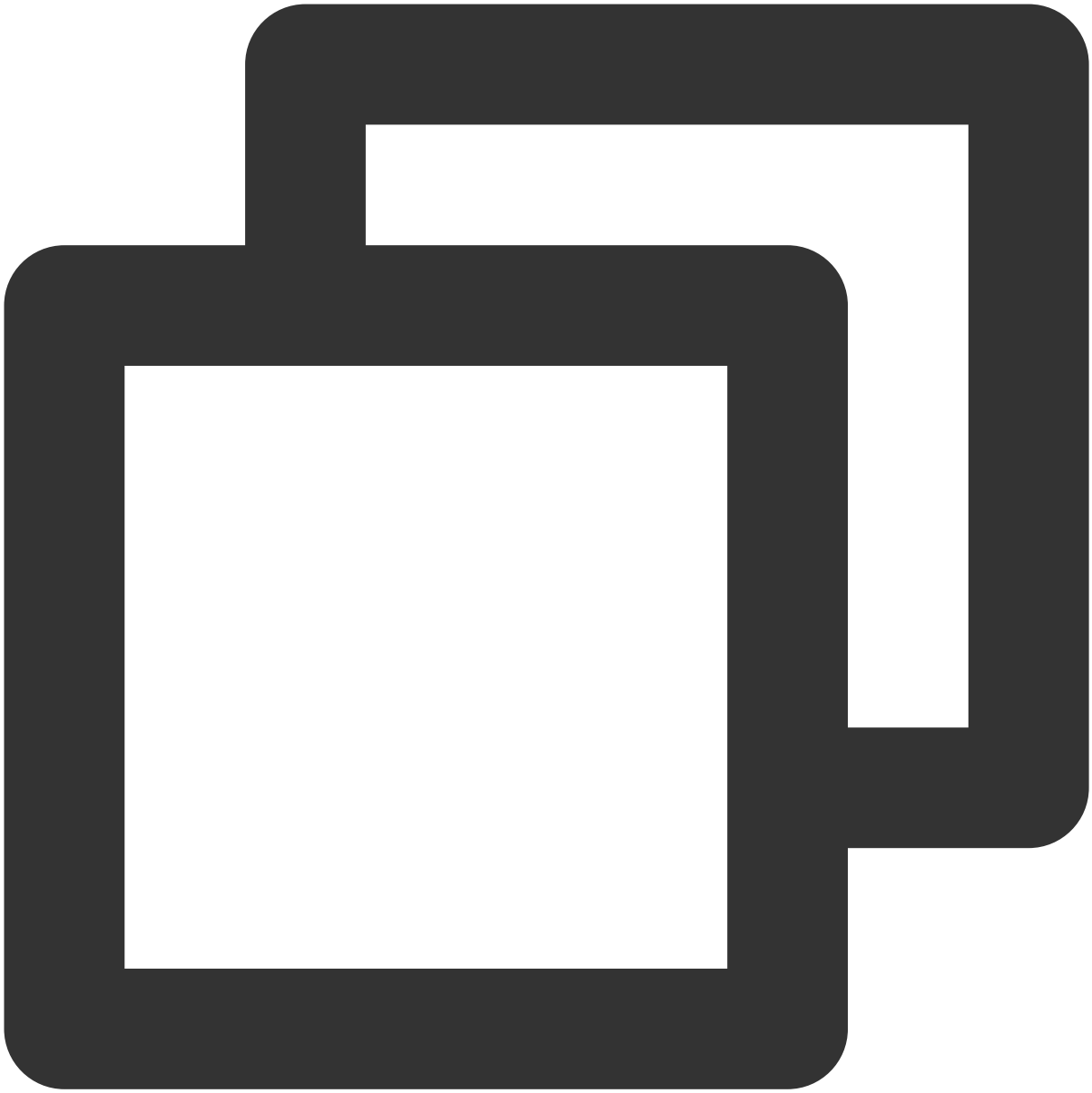
ここで、drupalはDrupalサービスが使用するデータベース名です。

4. データベースのユーザーを作成します。



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

ユーザーを承認し、承認が成功したらデータベースを終了します。

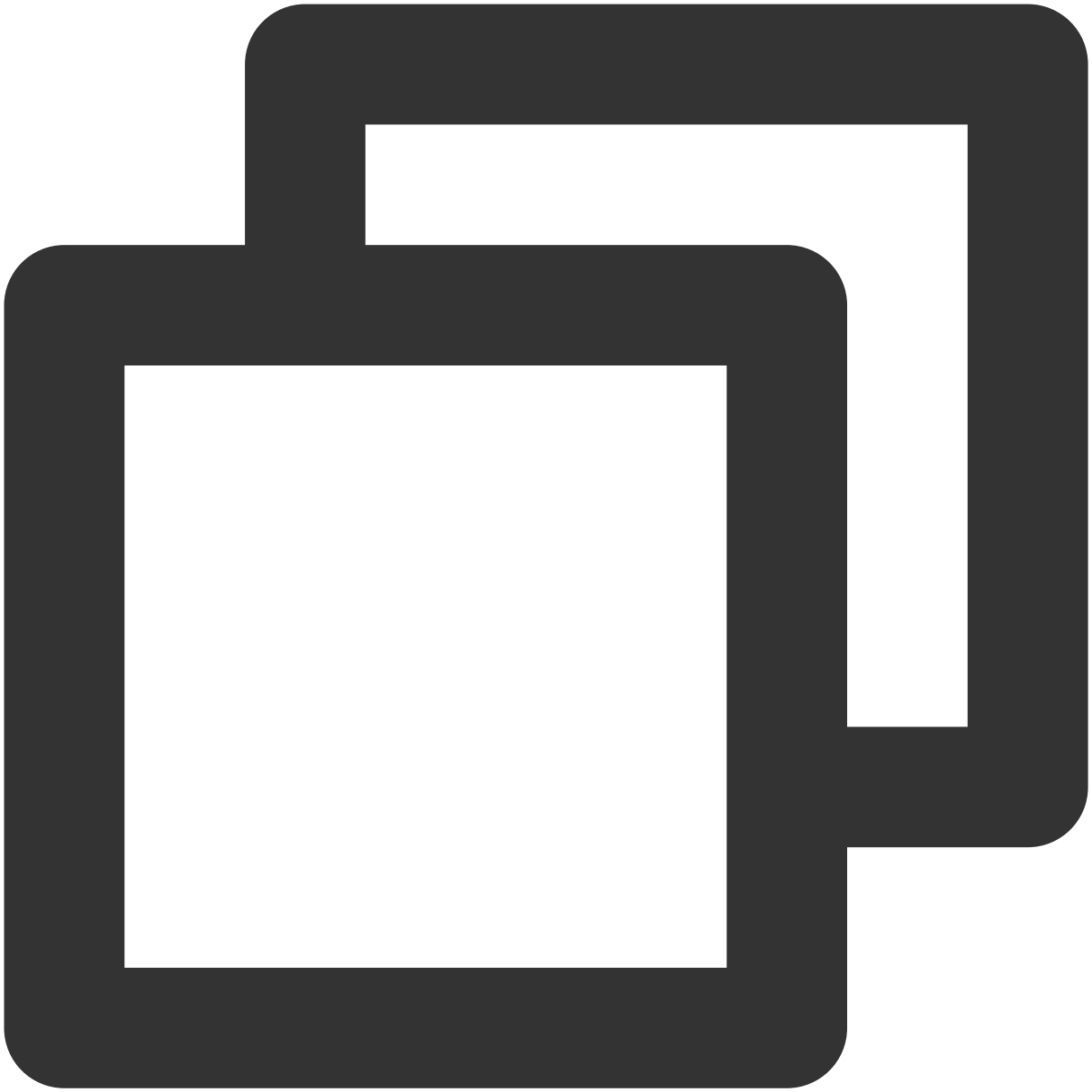


```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER, CREATE TEMPORARY
FLUSH PRIVILEGES;
exit
```

ここで、usernameはDrupalサービスが使用するデータベースのユーザー名で、passwordはDrupalサービスが使用するデータベースのパスワードです。

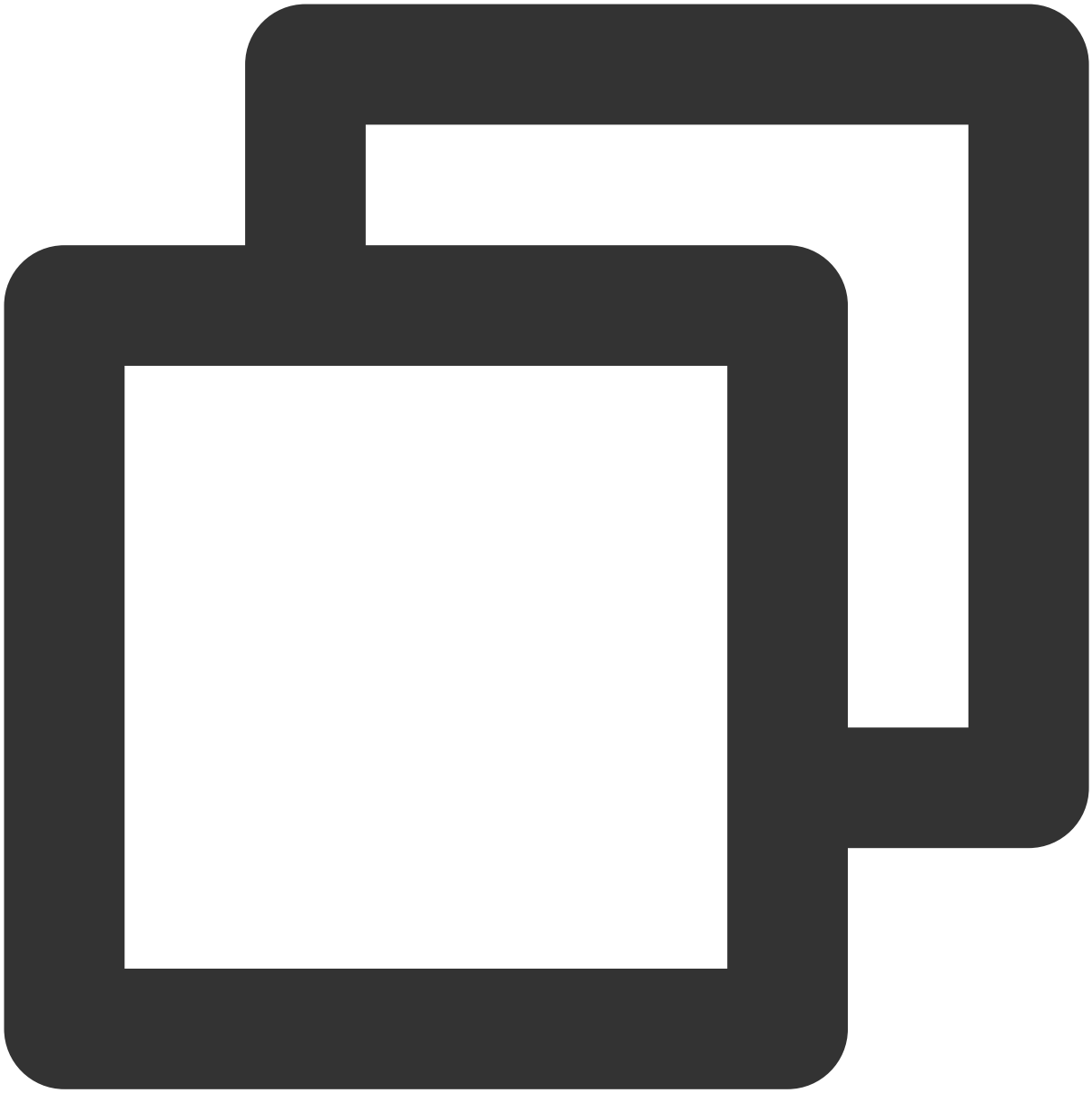
Apacheサービスのインストール

1. CVMインスタンスでは、 `yum` を使用してApacheをインストールします。



```
yum install httpd -y
```

2. Apacheサービスを起動します。



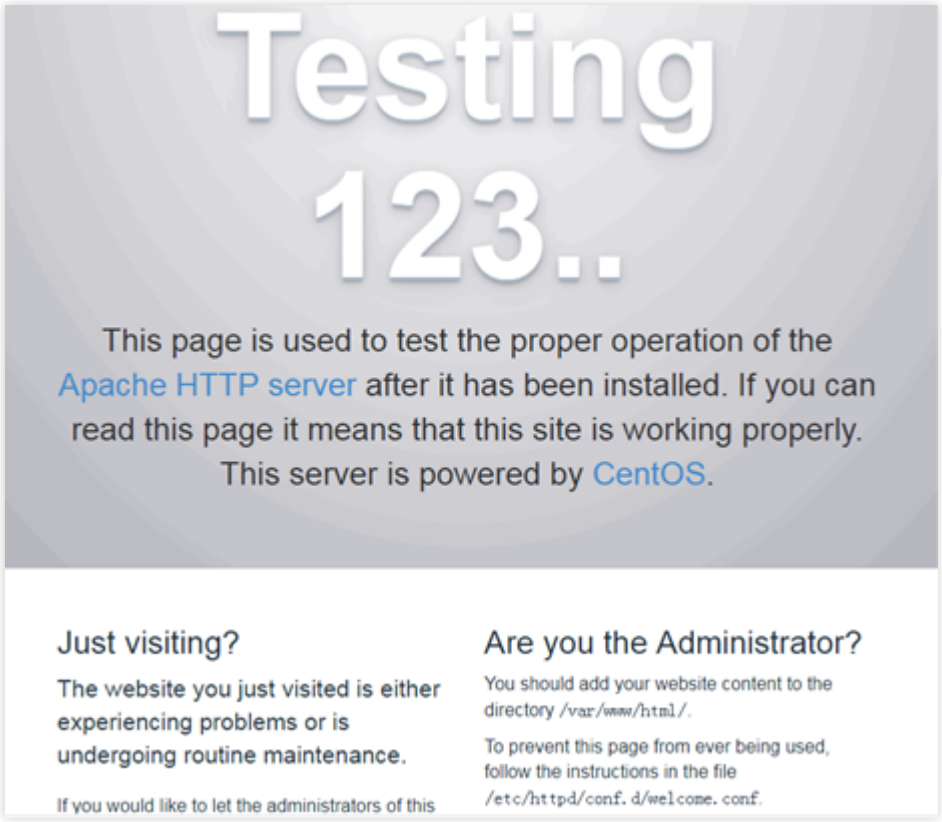
```
service httpd start
```

3. Apacheをテストします。

ご注意：

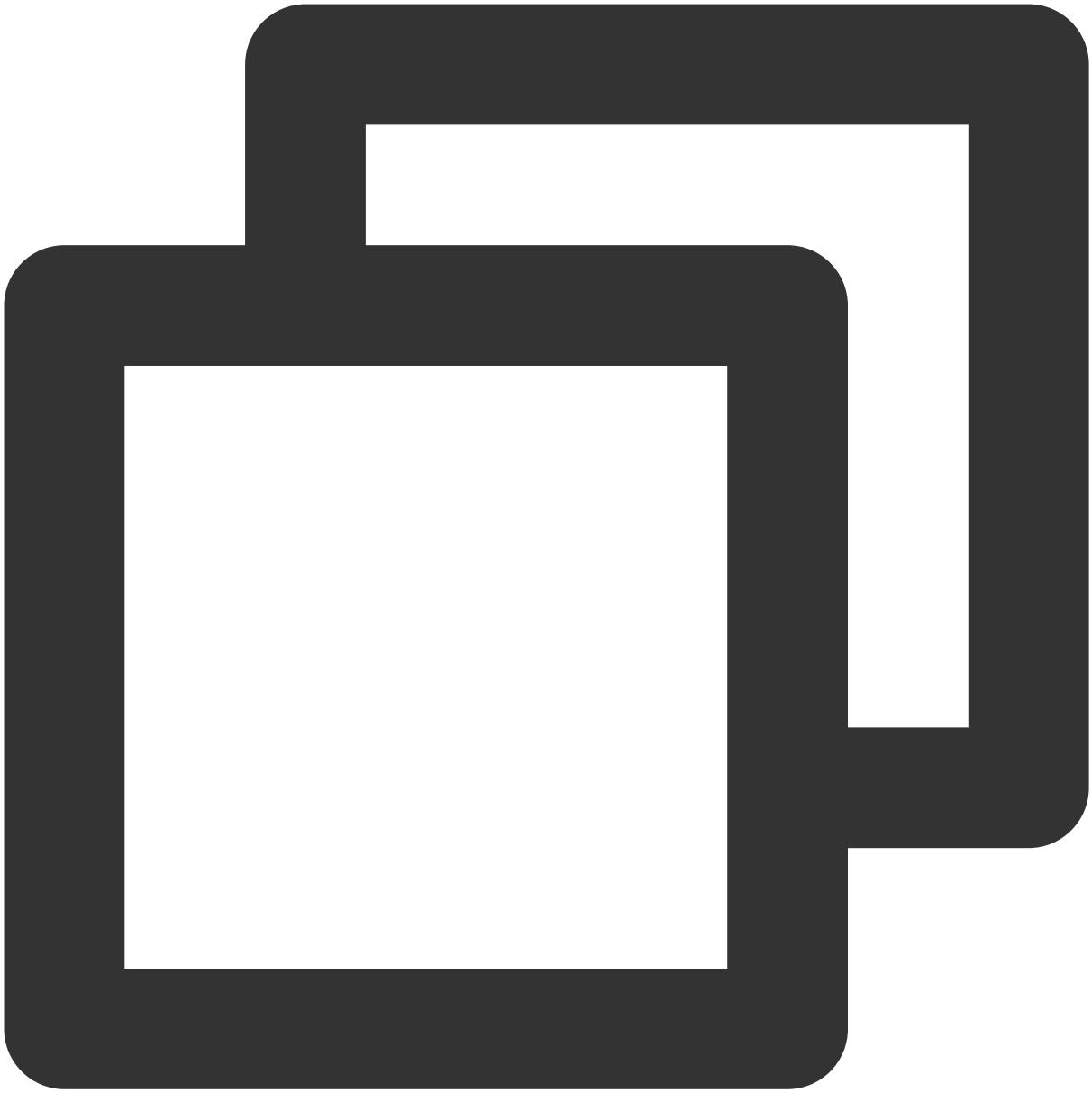
この手順では、CVMは、セキュリティグループでソースを**all**、ポートプロトコルを**TCP:80**とするインバウンドルールを構成する必要があります。セキュリティグループ構成については、セキュリティグループをご参照ください。

ローカルブラウザに `http://115.xxx.xxx.xxx/` （ここで、 `115.xxx.xxx.xxx` はCVMのパブリックネットワークIPアドレス）を入力すると、次の画面が表示され、Apacheが正常に起動しました。



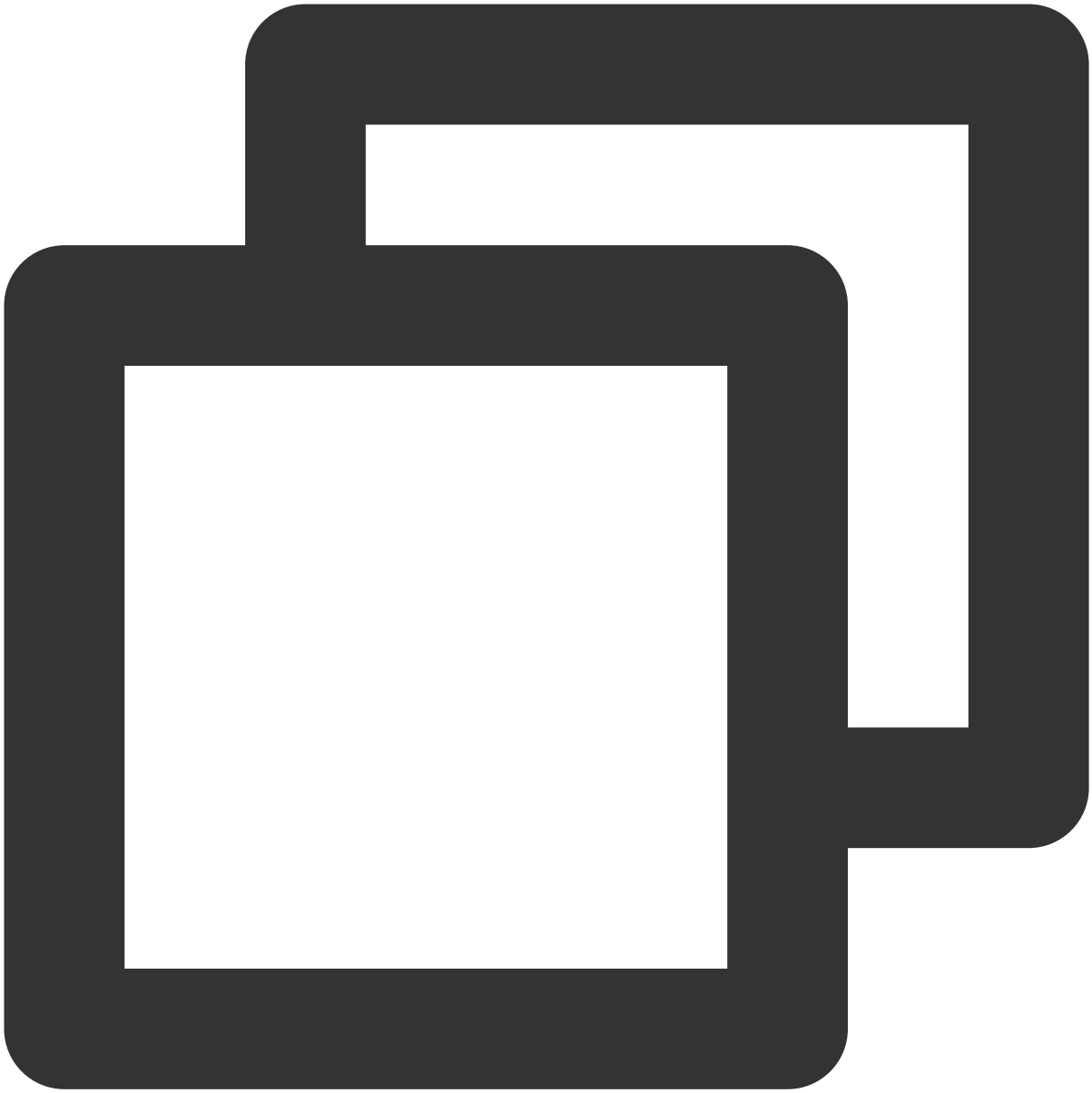
PHPのインストール

1. CVMインスタンスでは、 `yum` を使用してPHPおよびその拡張機能をインストールします。



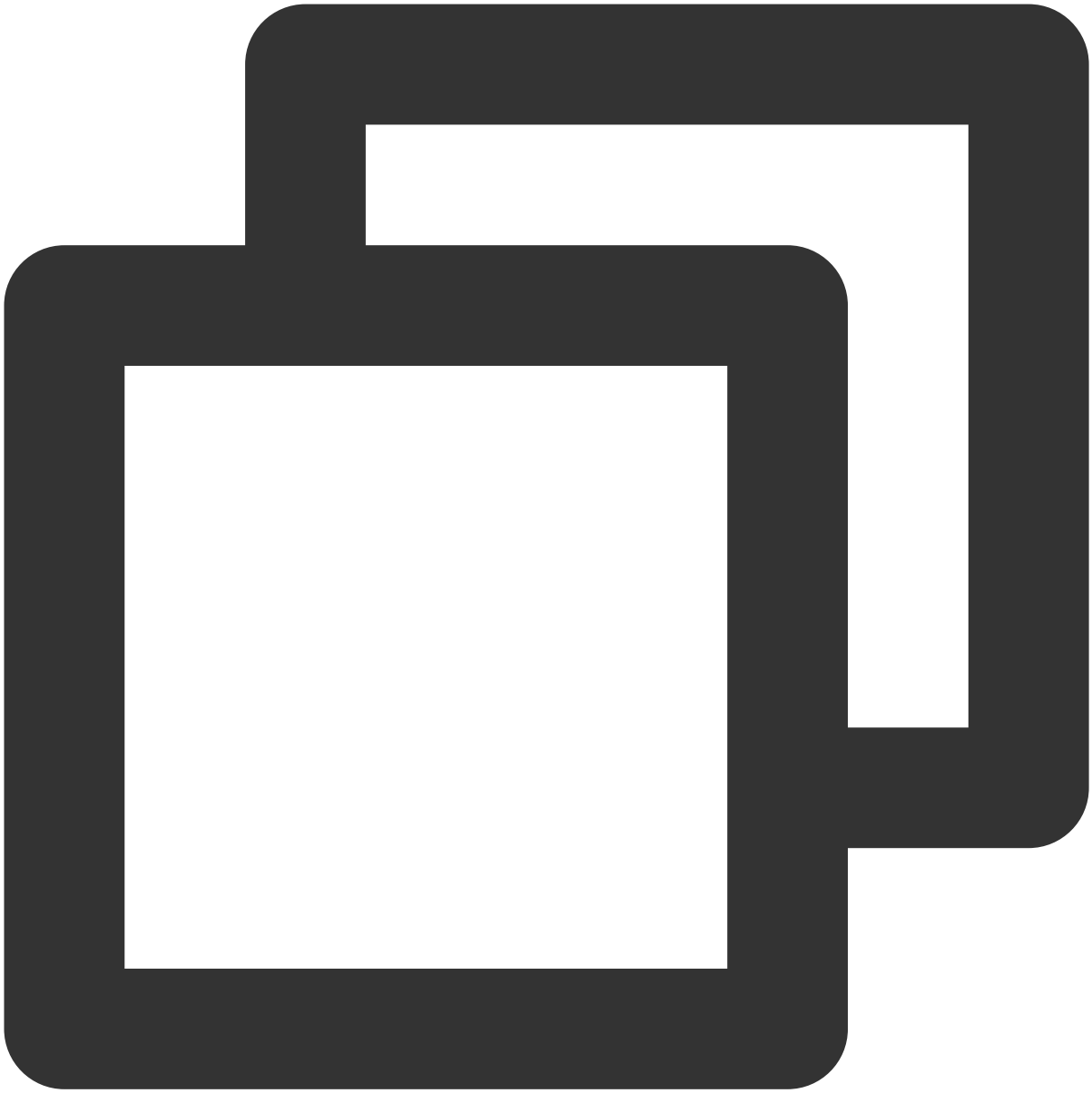
```
yum install php php-dom php-gd php-mysql php-pdo -y
```

2. CVMの `/var/www/html` ディレクトリ下で、info.phpフォルダを作成してPHPが正常にインストールされたかどうかを確認します。サンプルコードは次の通りです：



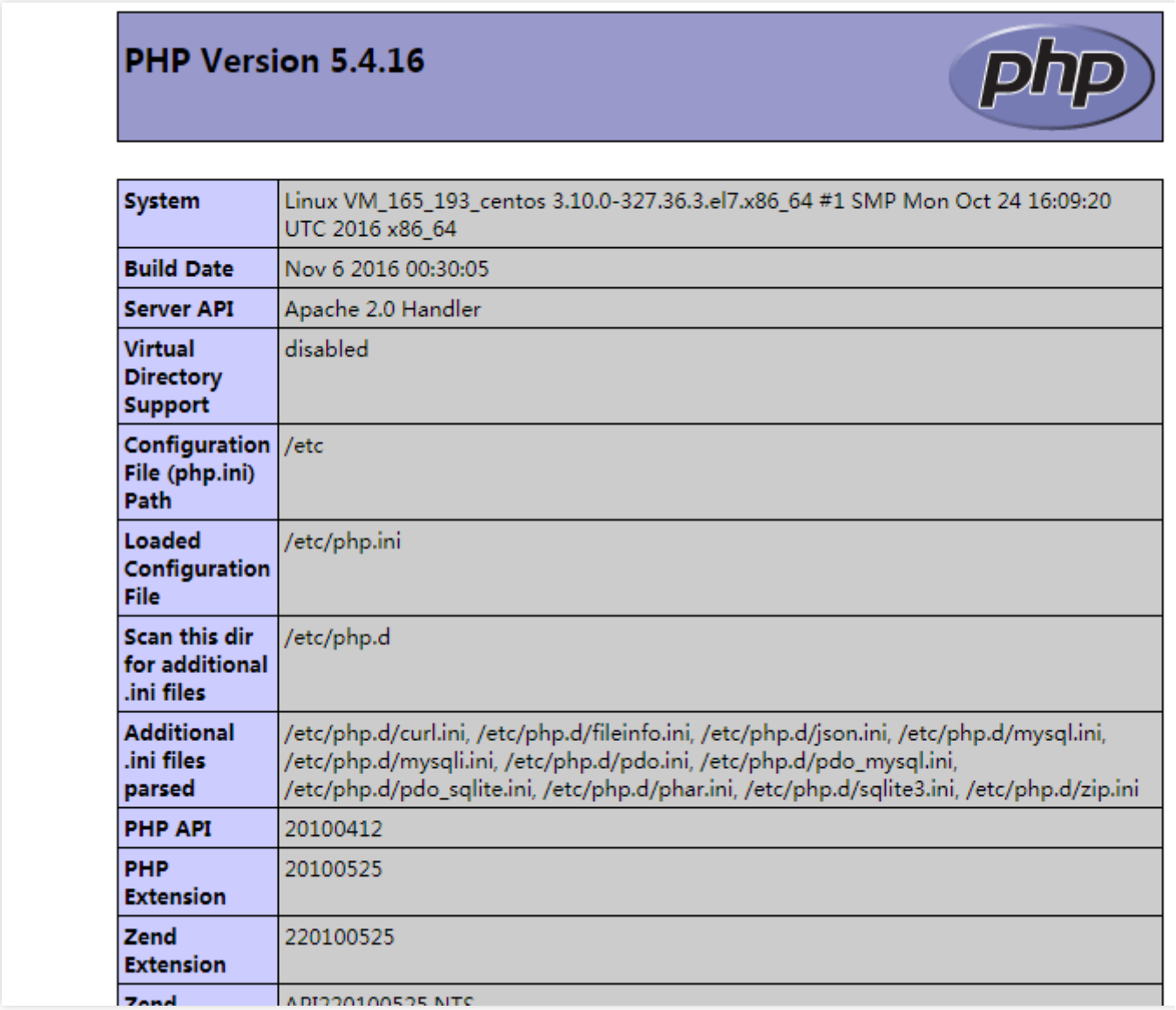
```
<?php phpinfo(); ?>
```

3. Apacheサービスを再起動します。



```
service httpd restart
```

4. ローカルブラウザに `http://115.xxx.xxx.xxx/info.php` （ここで、 `115.xxx.xxx.xxx` はCVMのパブリックネットワークIPアドレス）を入力すると、次の画面が表示され、PHPが正常にインストールされました。

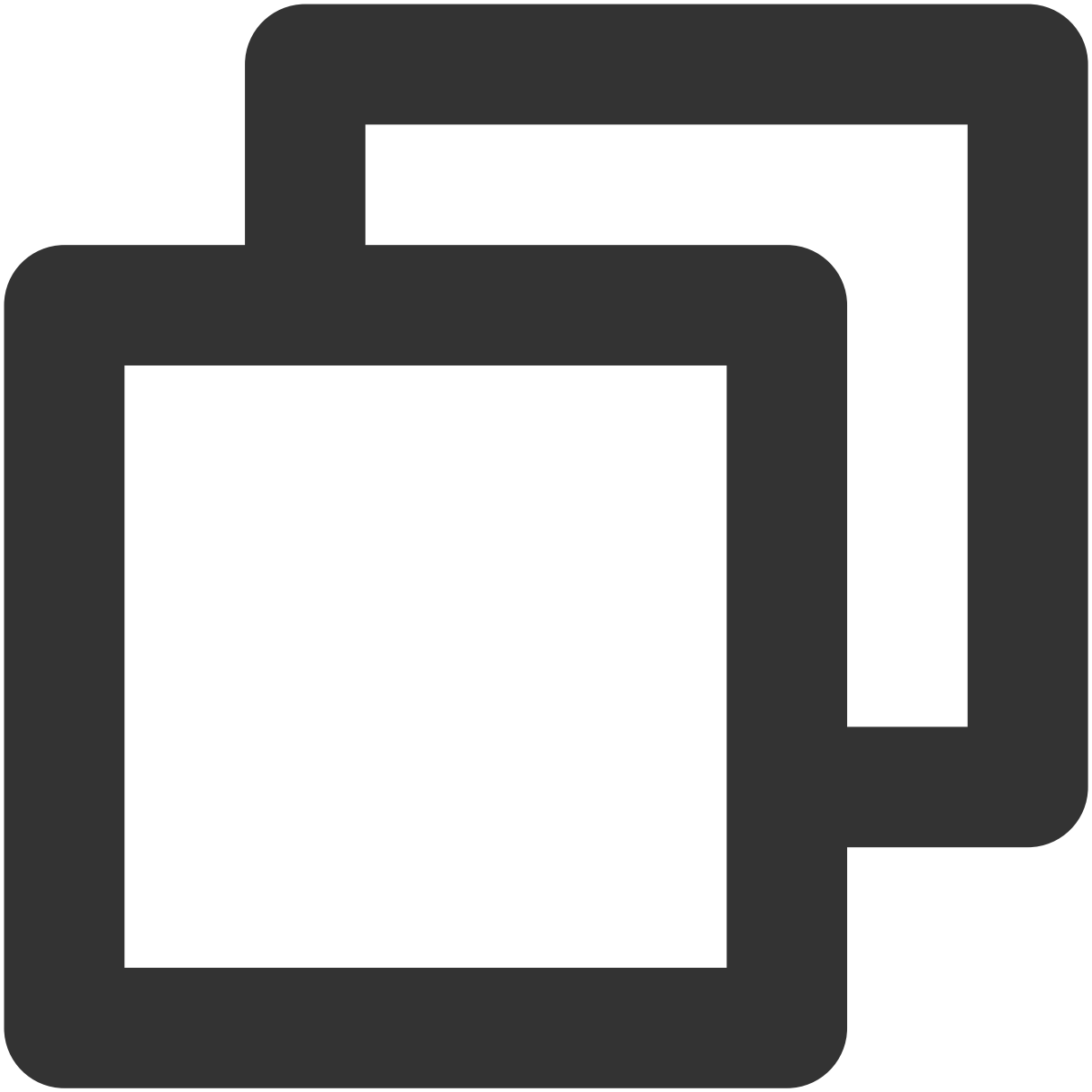


PHP Version 5.4.16

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

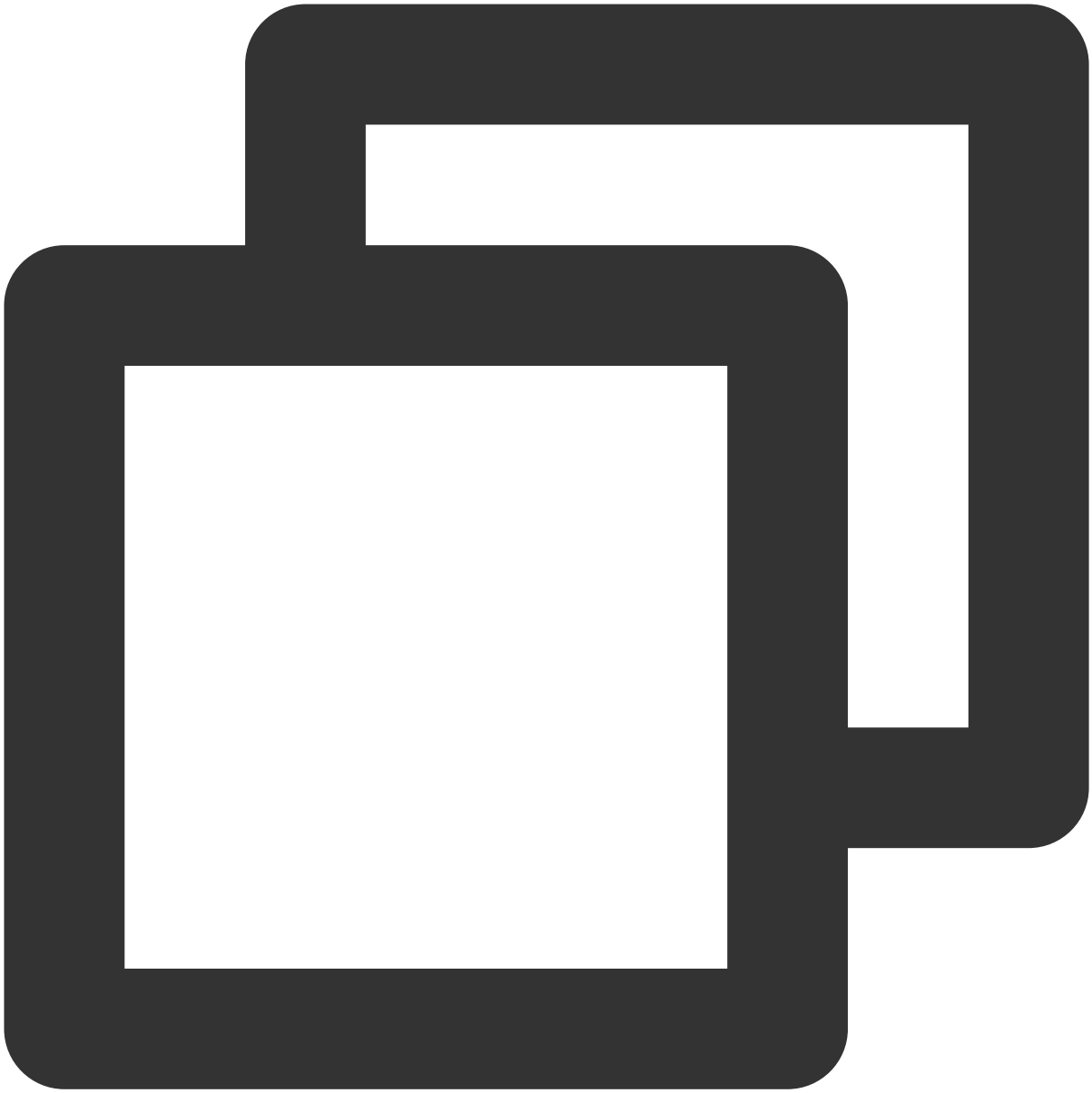
Drupalサービスのインストール

1. Drupalインストールパッケージをダウンロードします。



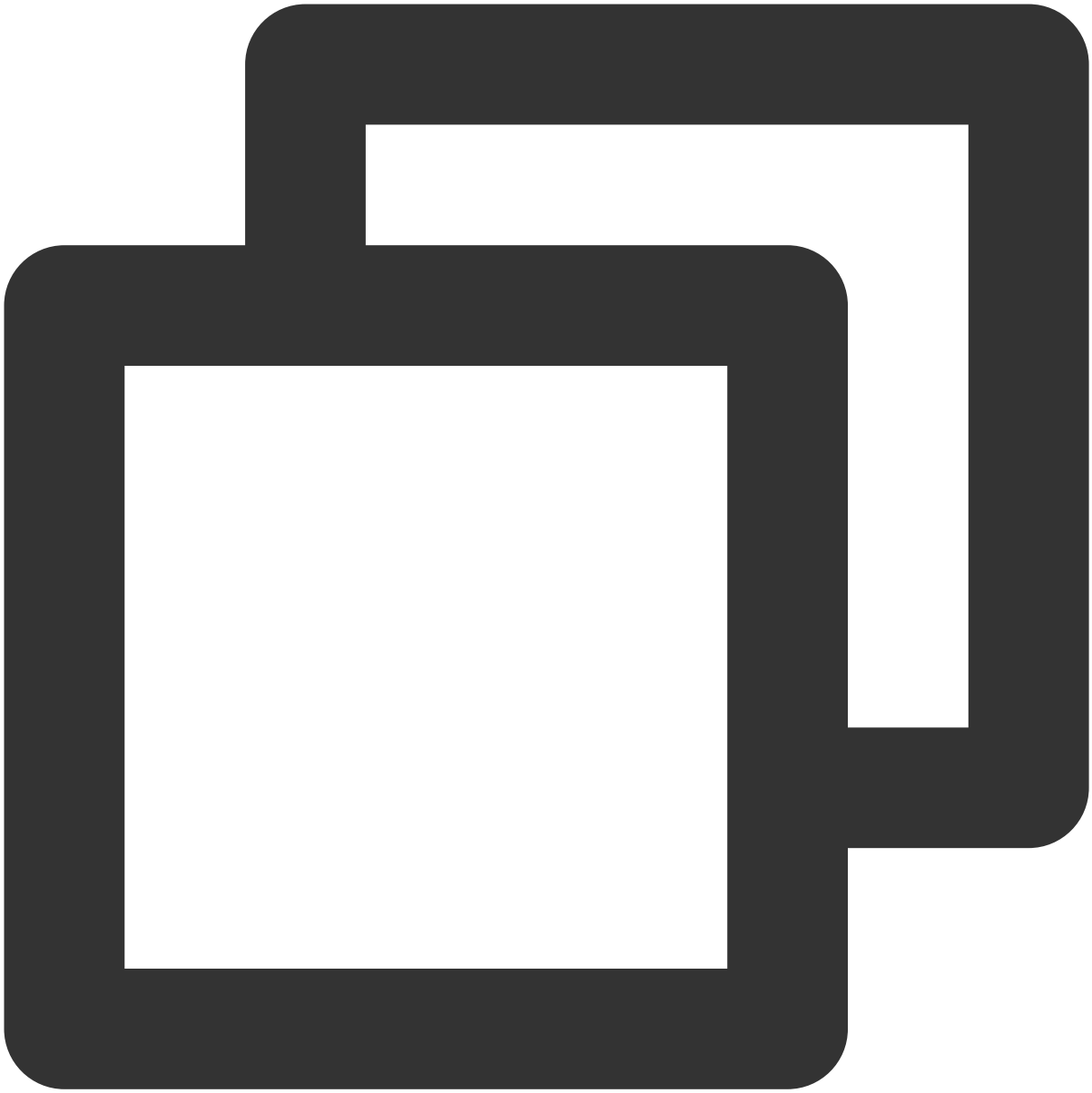
```
wget http://ftp.drupal.org/files/projects/drupal-7.56.zip
```

2. ウェブサイトのルートディレクトリに解凍します。



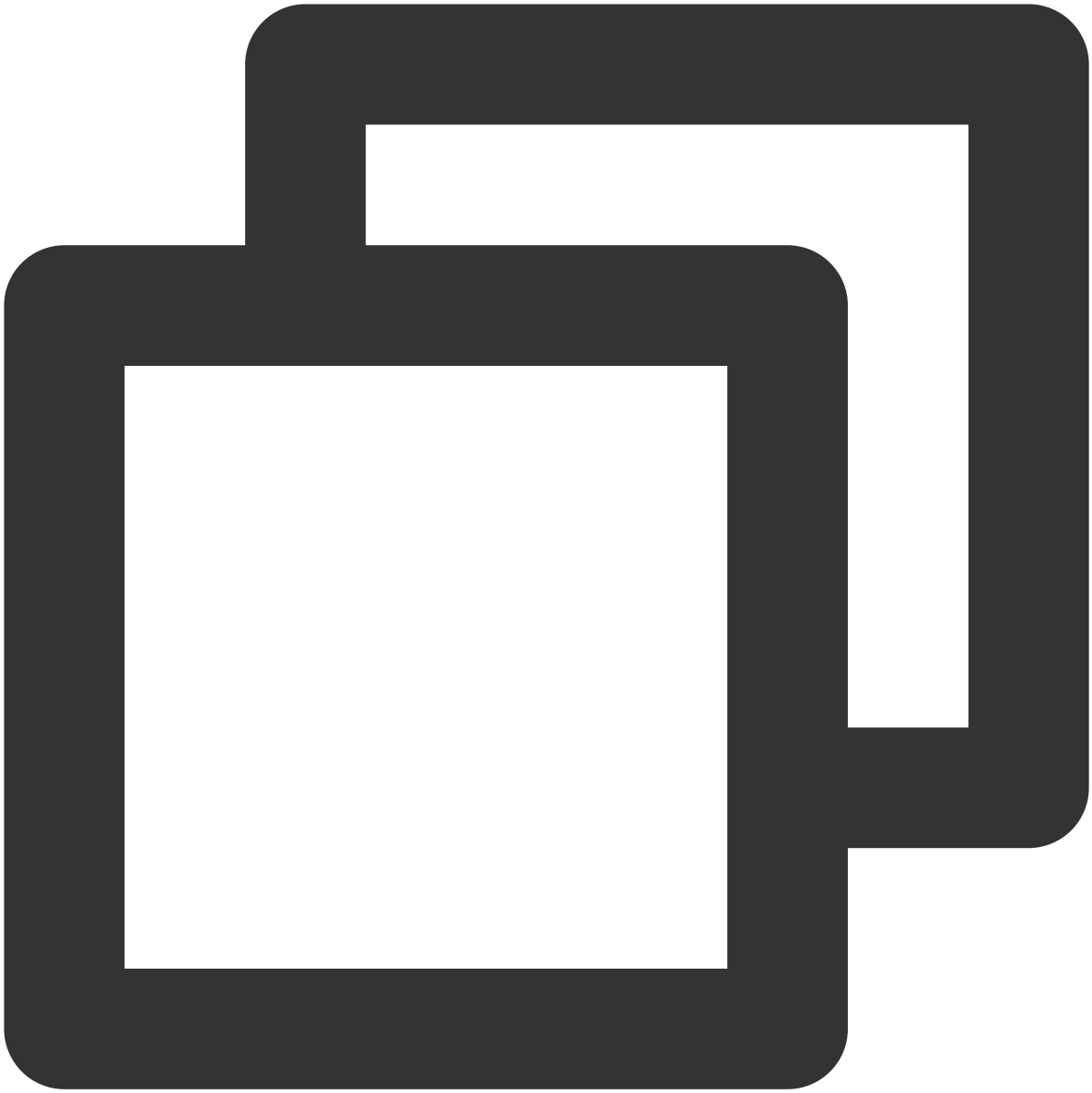
```
unzip drupal-7.56.zip
mv drupal-7.56/* /var/www/html/
```

3. 中国語翻訳パッケージをダウンロードします。



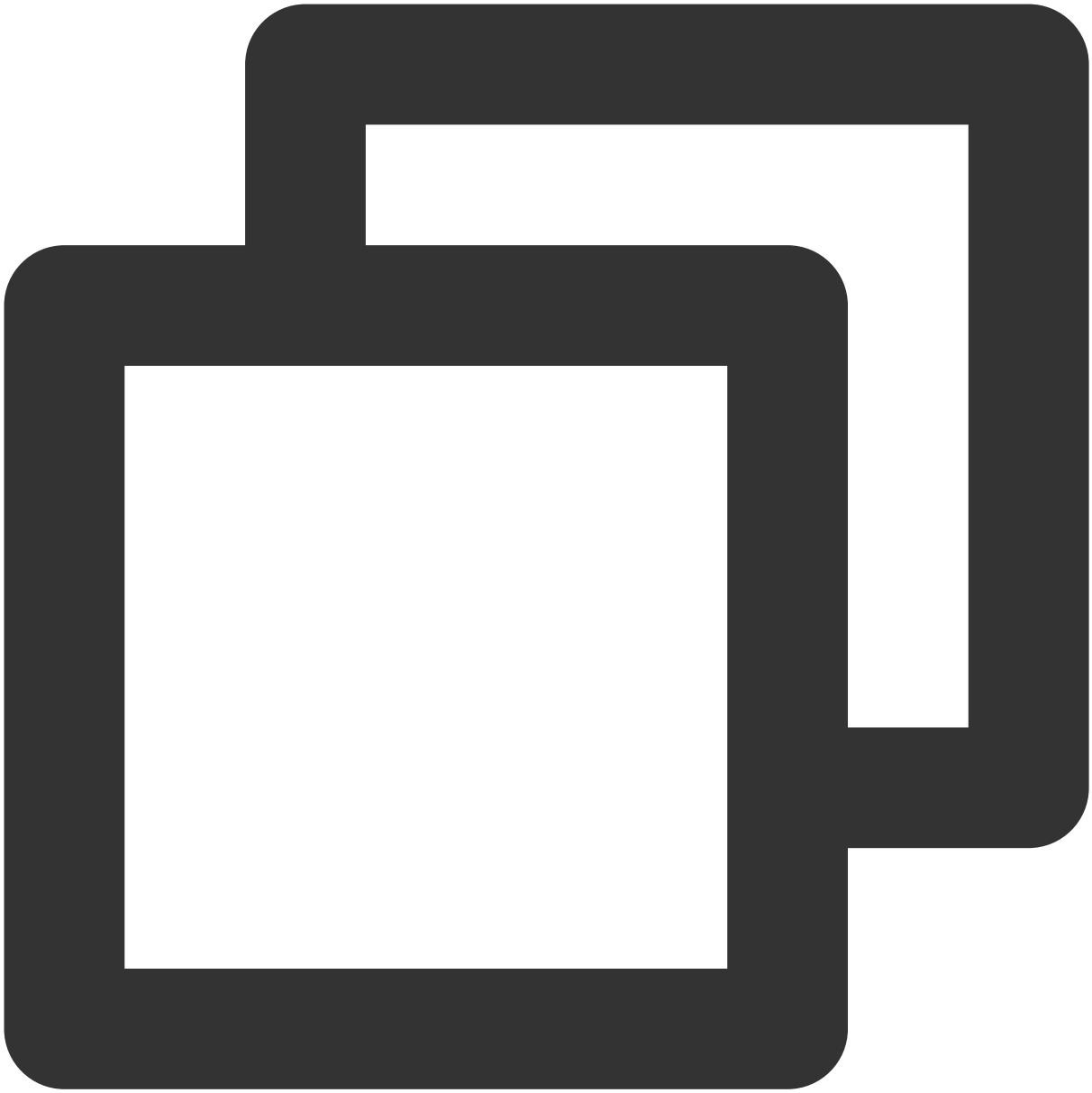
```
cd /var/www/html/
wget -P profiles/standard/translations http://ftp.drupal.org/files/translations/7.x
```

4. `sites` ディレクトリの所有者グループを変更します。



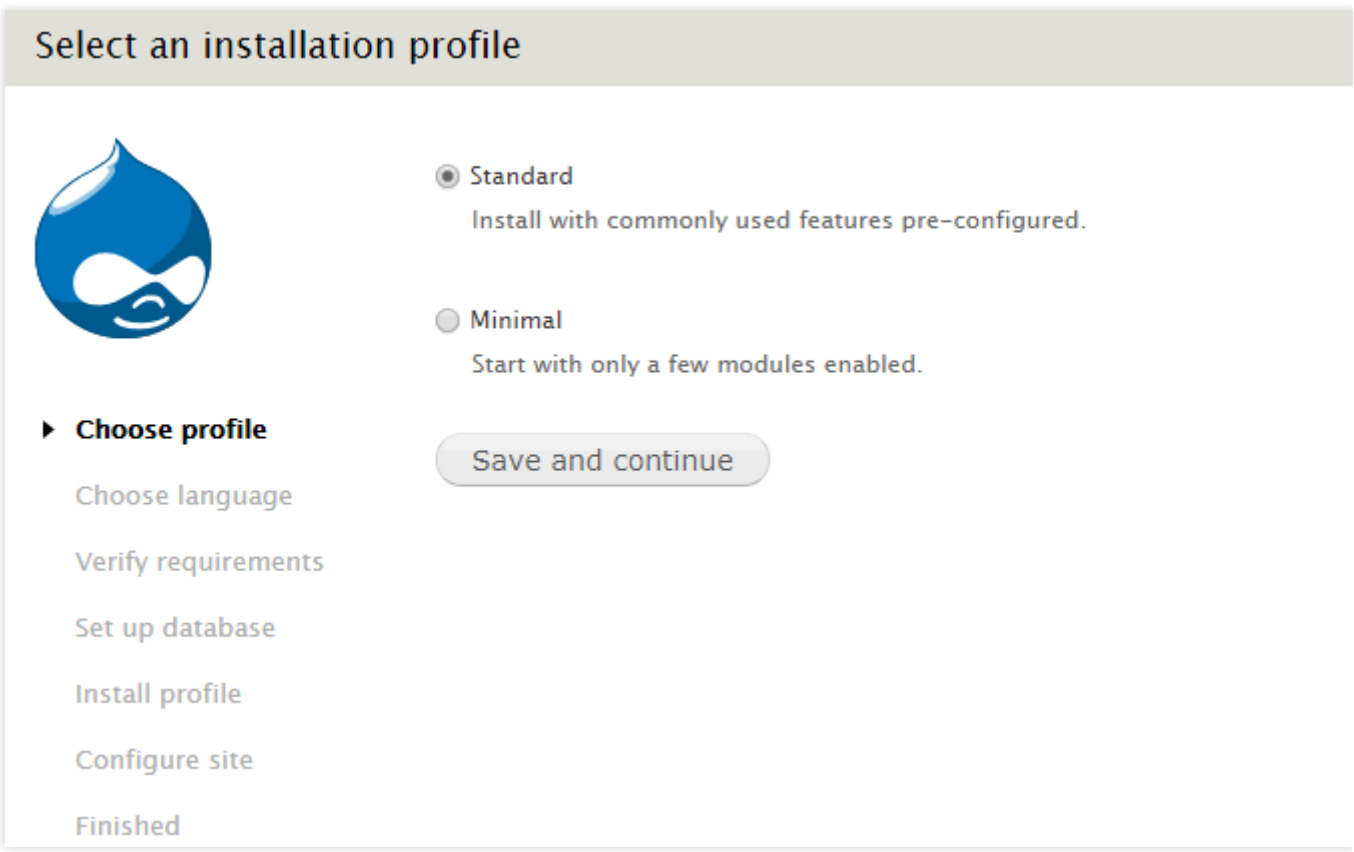
```
chown -R apache:apache /var/www/html/sites
```

5. Apacheサービスを再起動します。



```
service httpd restart
```

6. ローカルブラウザに `http://115.xxx.xxx.xxx/` （ここで、 `115.xxx.xxx.xxx` はCVMのパブリックネットワークIPアドレス）を入力すると、Drupalのインストールインターフェースに進みます。インストールするバージョンを選択し、【Save and continue】をクリックします。



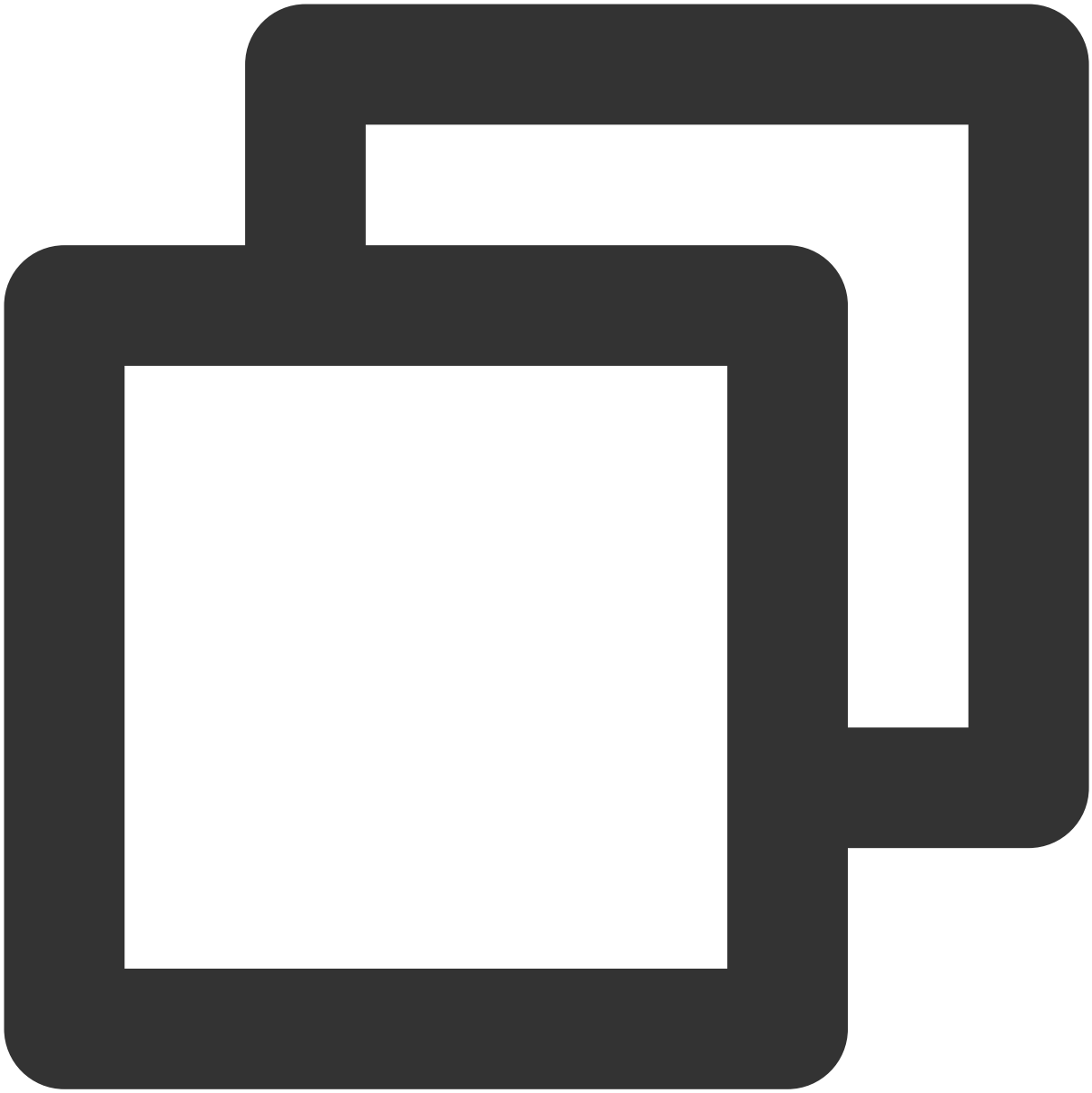
7. インストールする言語を選択し、【Save and continue】をクリックします。
8. データベースを設定し、**mariadbサービスのインストール**で構成したデータベース情報を入力します。
9. サイト情報を入力します。
10. Drupalのインストールを完了します。
11. その後で、`http://115.xxx.xxx.xxx/`（ここで、`115.xxx.xxx.xxx`はCVMのパブリックネットワークIPアドレス）にアクセスして、ウェブサイトをパーソナライズ設定することができます。

Python言語によるMySQL APIの使用
インスタンスの購入

最終更新日：：2024-07-25 17:56:18

API	説明
CreateDBInstance	TencentDBインスタンスを作成します
CreateDBInstanceHour	TencentDBインスタンス（従量課金）を作成します
DescribeDBInstances	インスタンスリストのクエリーを行います
DescribeDBPrice	データベース料金のクエリーを行います
DescribeDBZoneConfig	TencentDBの販売可能な仕様を取得します
InitDBInstances	新規インスタンスを初期化します

CreateDBInstance TencentDBインスタンスを作成します



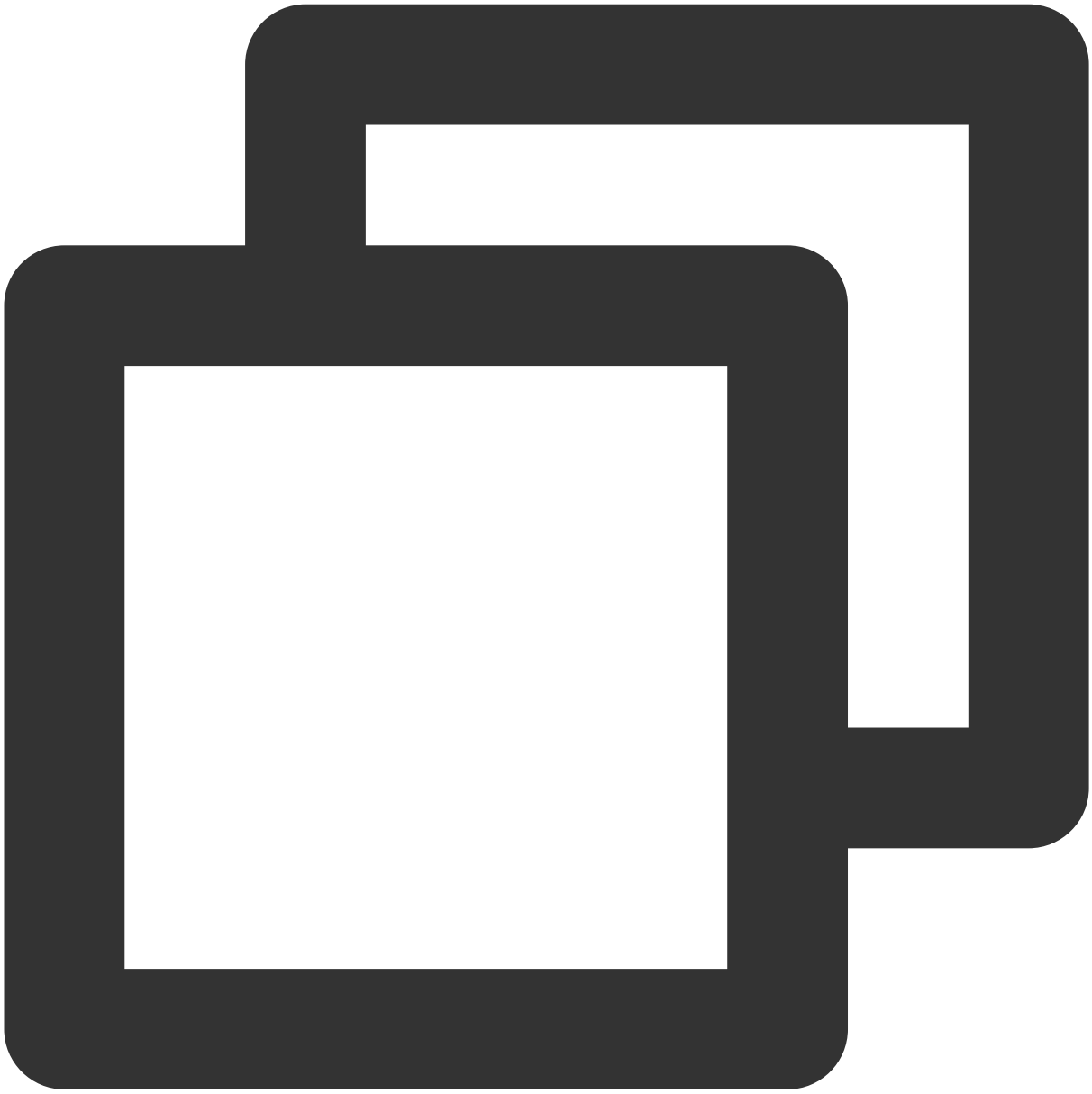
```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# TencentCloud APIのエントリーコンポーネントをインポートします
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudS
from tencentcloud.cdb.v20170320 import cdb_client, models
```

```
'''マスターインスタンスの購入'''
def CreateDBInstancedemomaster():
    try:
        # 認証オブジェクトをインスタンス化します。パラメータ入力には、Tencent Cloudアカウントの
        cred = credential.Credential("secretId", "secretKey")

        #製品（例はcdb）をリクエストするclientオブジェクトをインスタンス化します
        client = cdb_client.CdbClient(cred, "ap-beijing")

        #リクエストオブジェクトをインスタンス化します:req = models.ModifyInstanceParamReque
        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"]


        # clientオブジェクトでアクセスするインターフェースを呼び出すには、リクエストオブジェクト
        resp = client.CreateDBInstance(req)
        # json形式の文字列をパッケージに出力します
        print(resp.to_json_string())


    except TencentCloudSDKException as err:
        msg = traceback.format_exc() # メソッド1
        print (msg)




'''読み取り専用インスタンスの購入'''
def CreateDBInstancedemoro():
    try:
        # 認証オブジェクトをインスタンス化します。パラメータ入力には、Tencent Cloudアカウントの
        cred = credential.Credential("secretId", "secretId")

        #製品（例はcdb）をリクエストするclientオブジェクトをインスタンス化します
        client = cdb_client.CdbClient(cred, "ap-beijing")
```

```
        #リクエストオブジェクトをインスタンス化します:req = models.ModifyInstanceParamReque
        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_c
        req.BackupZone = "0"
        req.AutoRenewFlag =0
        req.MasterInstanceId ="cdb-bgr97hu0"
        req.RoGroup = {"RoGroupMode":"allinone","RoGroupName":"roweek"}
        req.InstanceName = "tencentcdbRO"


        # clientオブジェクトでアクセスするインターフェースを呼び出すには、リクエストオブジェクト
        resp = client.CreateDBInstance(req)
        # json形式の文字列をパッケージに出力します
        print(resp.to_json_string())


    except TencentCloudSDKException as err:
        msg = traceback.format_exc() # メソッド1
        print (msg)


'''ディザスタリカバリインスタンスの購入'''
def CreateDBInstancedemodr():
    try:
        # 認証オブジェクトをインスタンス化します。パラメータ入力には、Tencent Cloudアカウントの
        cred = credential.Credential("secretId", "secretKey")

        #製品（例はcdb）をリクエストするclientオブジェクトをインスタンス化します
        client = cdb_client.CdbClient(cred, "ap-shanghai")

        #リクエストオブジェクトをインスタンス化します:req = models.ModifyInstanceParamReque
        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_c
        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,M
        #req.RoGroup = [roGroup]
        req.InstanceName = "tencentcdbDR"


        # clientオブジェクトでアクセスするインターフェースを呼び出すには、リクエストオブジェクト
        resp = client.CreateDBInstance(req)
        # json形式の文字列をパッケージに出力します
        print(resp.to_json_string())


    except TencentCloudSDKException as err:
        msg = traceback.format_exc() # メソッド1
        print (msg)

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

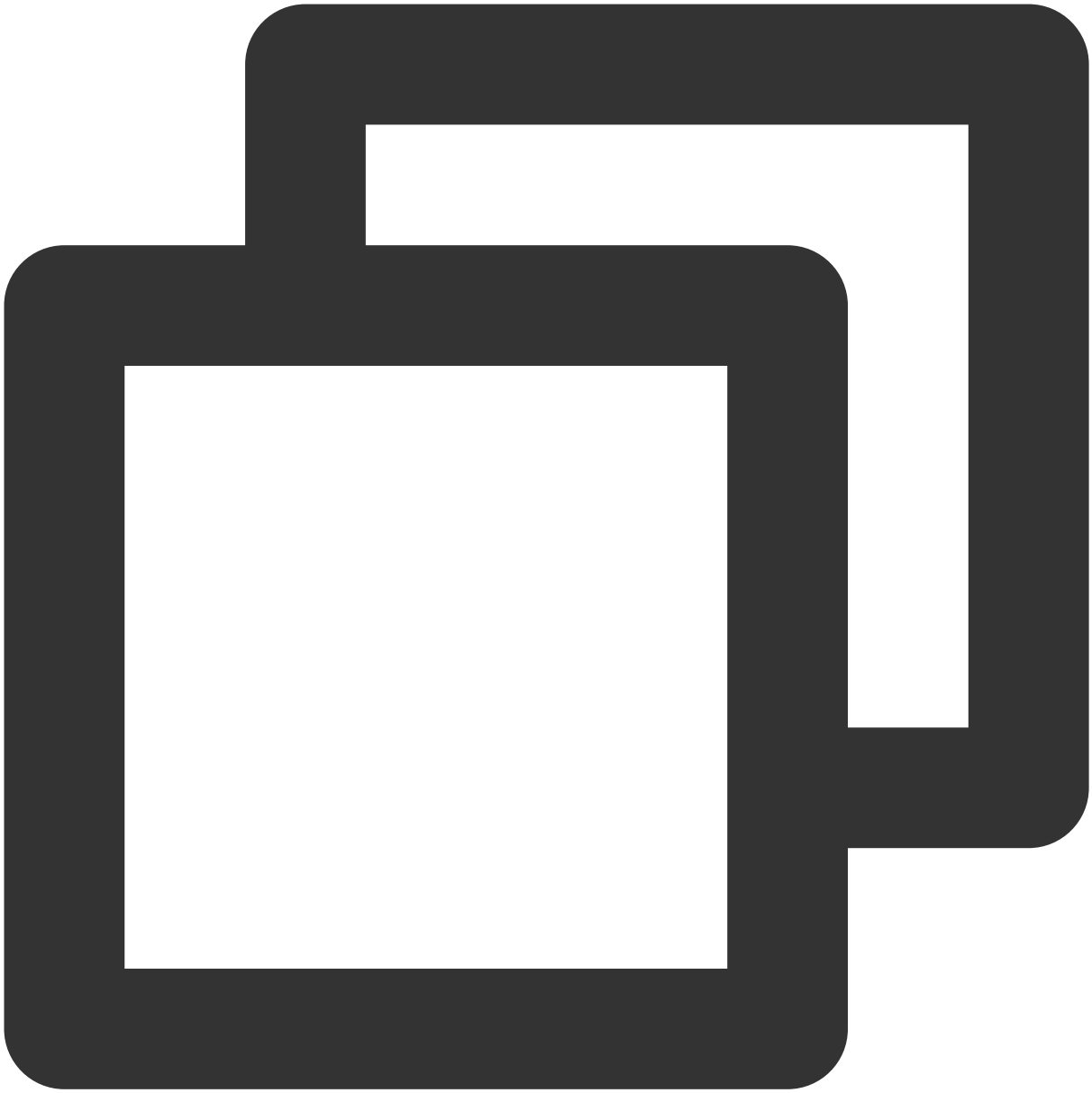
CreateDBInstanceHour TencentDBインスタンス（従量課金）を作成します



```
'''1時間単位の課金の場合、凍結された金額が必要で、アカウントに残額が必要です。アカウント残額が0の場
#!/usr/bin/python
# -*- coding: utf-8 -*-

# TencentCloud APIのエントリーコンポーネントをインポートします
import logging
```

```
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudS
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # 認証オブジェクトをインスタンス化します。パラメータ入力には、Tencent Cloudアカウントのsecr
    cred = credential.Credential("secretId", "secretKey")

    #製品（例はcdb）をリクエストするclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-beijing")

    #リクエストオブジェクトをインスタンス化します: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_
    req.ProtectMode = 1
    req.SlaveZone = "ap-beijing-3"
    req.InstanceName = "oneday1"
    req.AutoRenewFlag = 0


    # clientオブジェクトでアクセスするインターフェースを呼び出すには、リクエストオブジェクトを渡す
    resp = client.CreateDBInstanceHour(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    msg = traceback.format_exc() # メソッド1
    print (msg)
```

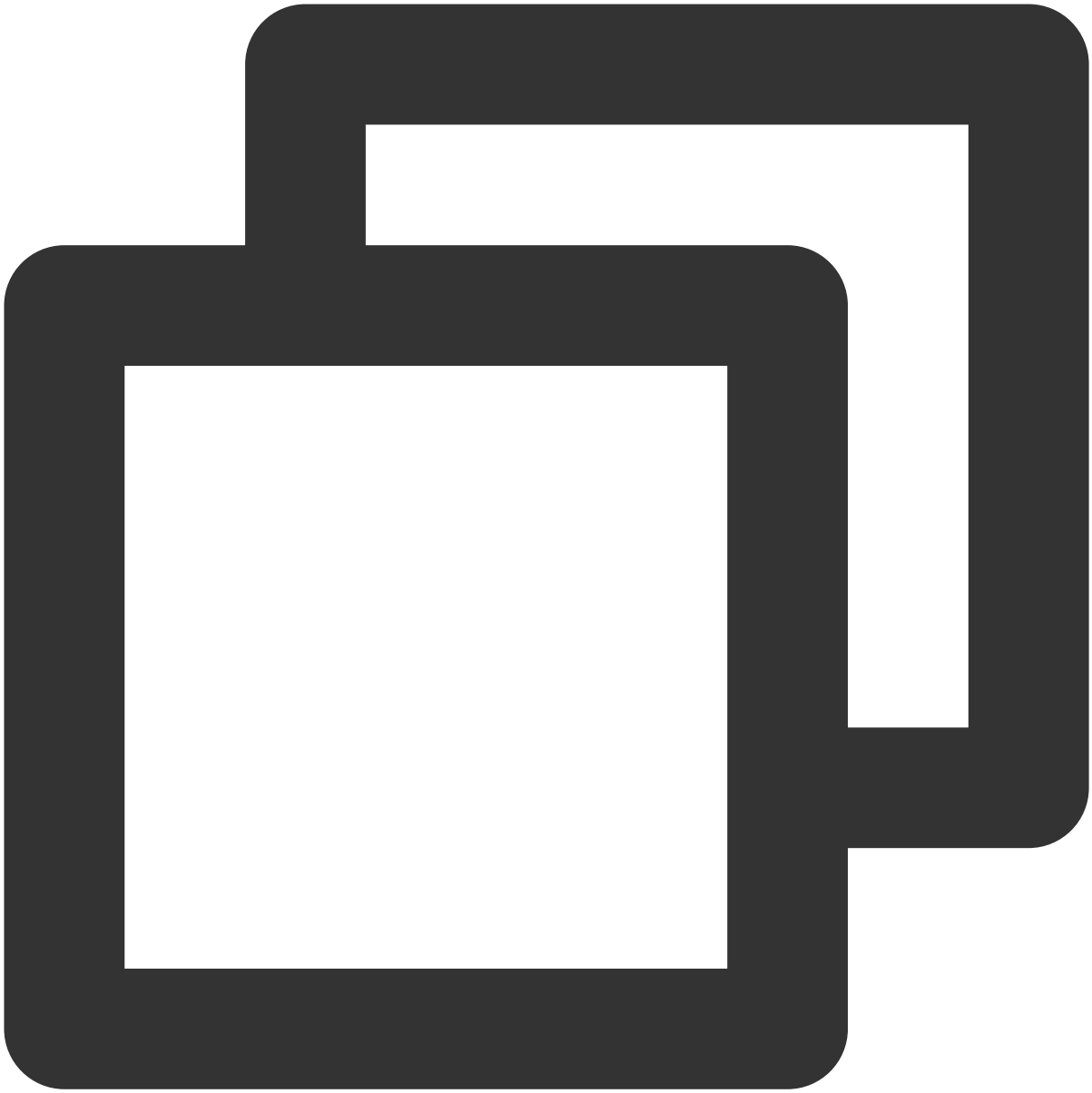
DescribeDBInstancesインスタンスリストのクエリーを行います



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

# TencentCloud APIのエントリーコンポーネントをインポートします
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudS
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
```

```
    # 認証オブジェクトをインスタンス化します。パラメータ入力には、Tencent Cloudアカウントのsecr
    cred = credential.Credential("secretId", "secretKey")

    #製品（例はcdb）をリクエストするclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #リクエストオブジェクトをインスタンス化します:req = models.ModifyInstanceParamRequest(
    req = models.DescribeDBInstancesRequest()
    req.EngineVersions = ["5.6"]
    req.OrderBy = "instanceId"
    req.InstanceIds = ["cdb-1j8lumf6"]


    # clientオブジェクトでアクセスするインターフェースを呼び出すには、リクエストオブジェクトを渡す
    resp = client.DescribeDBInstances(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
        msg = traceback.format_exc() # メソッド1
        print (msg)
```

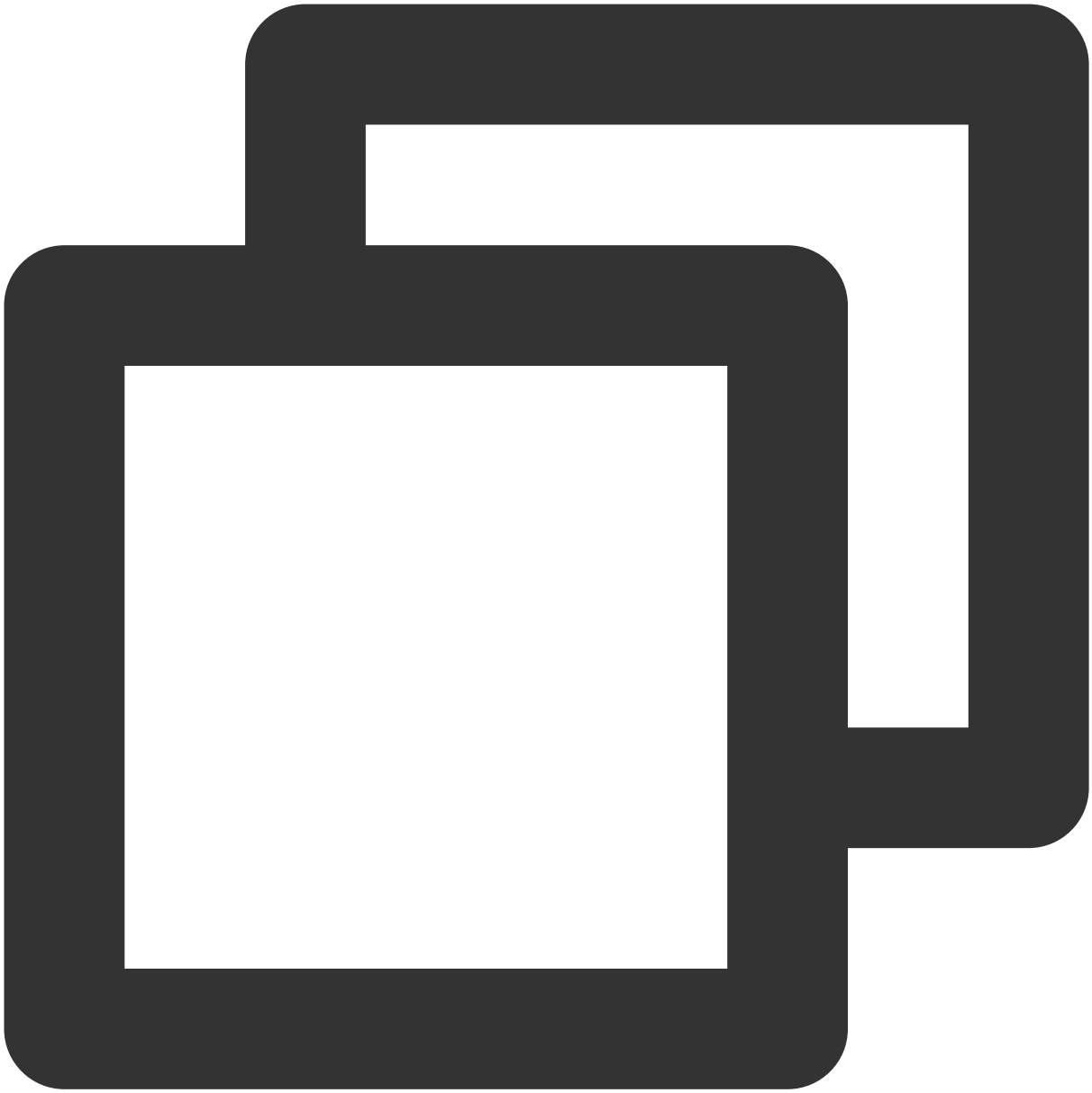
DescribeDBPrice データベースの価格のクエリーを行います



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

# TencentCloud APIのエントリーコンポーネントをインポートします
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudS
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # 認証オブジェクトをインスタンス化します。パラメータ入力には、Tencent Cloudアカウントのsecr
    cred = credential.Credential("secretId", "secretKey")
```

```
    #製品（例はcdb）をリクエストするclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-guangzhou")

    #リクエストオブジェクトをインスタンス化します: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


    # clientオブジェクトでアクセスするインターフェースを呼び出すには、リクエストオブジェクトを渡す
    resp = client.DescribeDBPrice(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

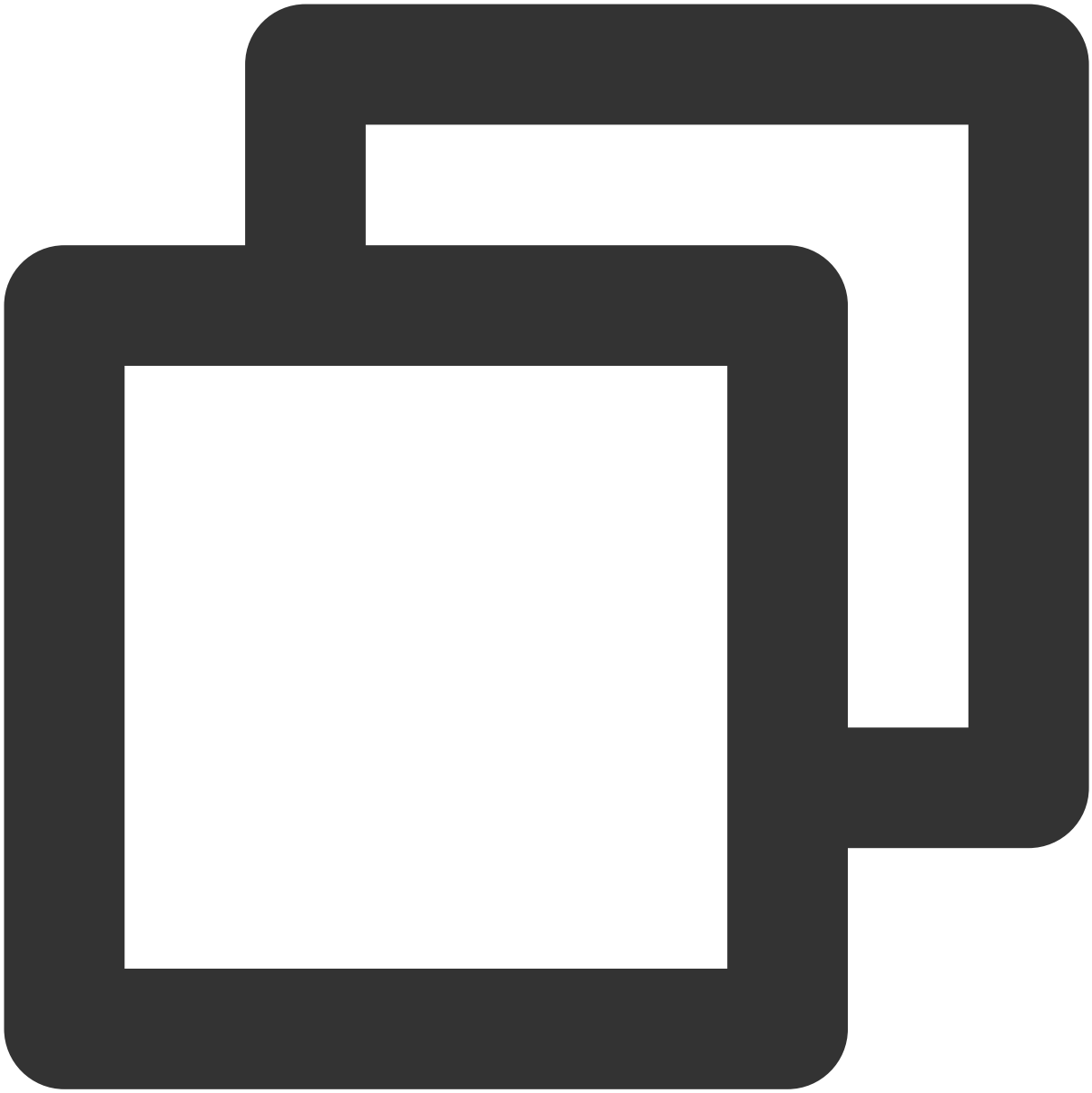
DescribeDBZoneConfig TencentDBの販売可能な仕様を取得します



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

# TencentCloud APIのエントリーコンポーネントをインポートします

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

try:
    # 認証オブジェクトをインスタンス化します。パラメータ入力には、Tencent Cloudアカウントのsecr
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例はcdb）をリクエストするclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #リクエストオブジェクトをインスタンス化します:req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBZoneConfigRequest()
    #req.InstanceId = "cdb-j0edpju5"


    # clientオブジェクトでアクセスするインターフェースを呼び出すには、リクエストオブジェクトを渡す
    resp = client.DescribeDBZoneConfig(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

InitDBInstances 新規インスタンスを初期化します



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

# TencentCloud APIのエントリーコンポーネントをインポートします

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

try:
```

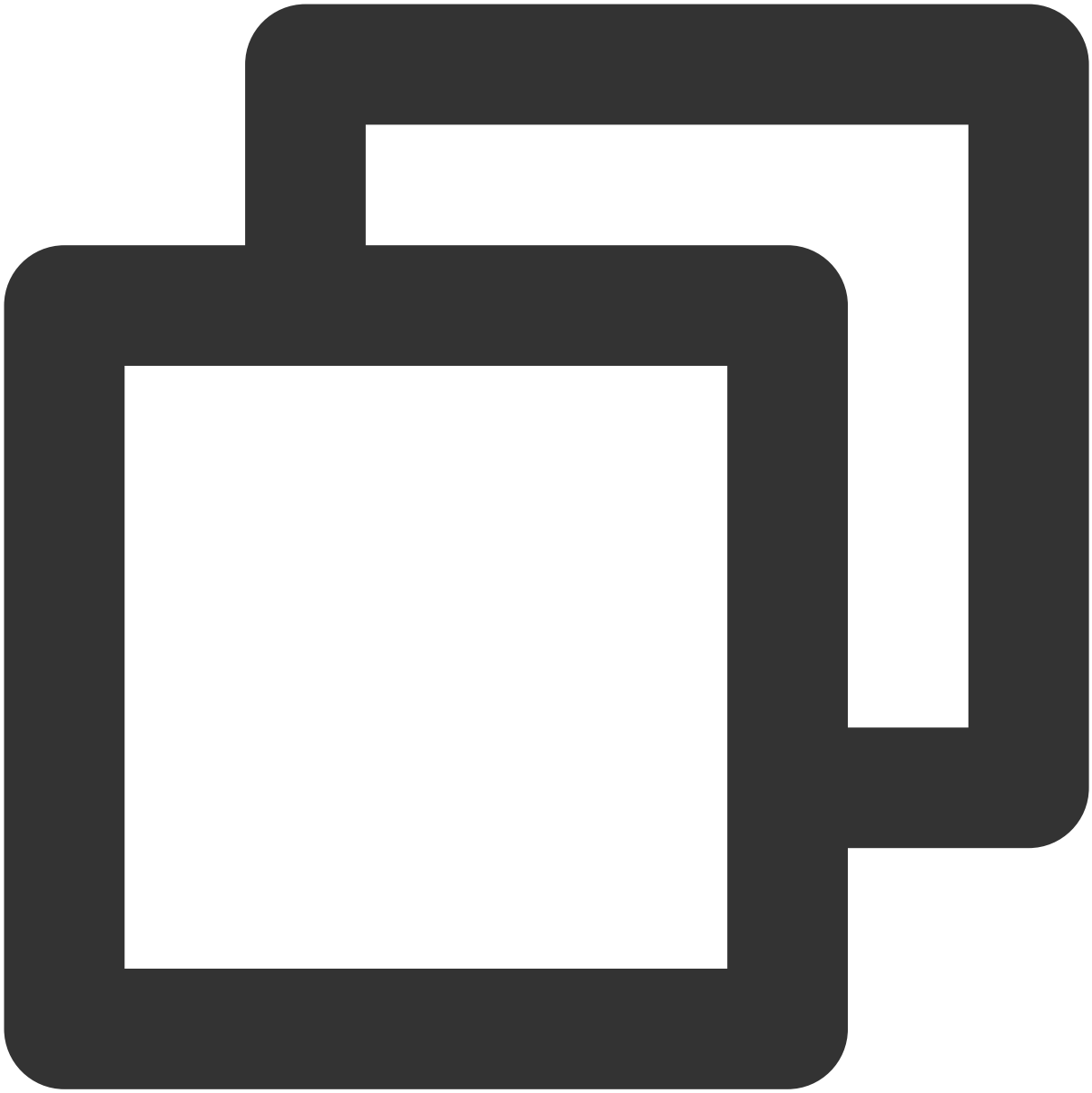
```
    # 認証オブジェクトをインスタンス化します。パラメータ入力には、Tencent Cloudアカウントのsecr
    cred = credential.Credential("secretId", "secretKey")

    #製品（例はcdb）をリクエストするclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #リクエストオブジェクトをインスタンス化します:req = models.ModifyInstanceParamRequest()
    req = models.InitDBInstancesRequest()
    req.InstanceIds = ["cdb-c752yqcn"]
    req.NewPassword = "CDB@Qcloud"

    req.Parameters = [{"name":"max_connections","value":"100"},{"name":"character_s


    # clientオブジェクトでアクセスするインターフェースを呼び出すには、リクエストオブジェクトを渡す
    resp = client.InitDBInstances(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

インスタンス管理

最終更新日：：2024-07-25 17:56:18

API	説明
ModifyInstanceParam	インスタンスパラメータを変更する
CloseWanService	インスタンスのインターネットアクセスを無効にする
OpenWanService	インスタンスのインターネットアクセスを有効にする
RestartDBInstances	インスタンスを再起動する
OpenDBInstanceGTID	インスタンスのGTIDを有効にする
ModifyDBInstanceName	MySQLのインスタンス名を変更する
ModifyDBInstanceProject	MySQLインスタンスの所属プロジェクトを変更する
ModifyDBInstanceVipVport	MySQLインスタンスのIPとポート番号を変更する
DescribeDBInstanceCharset	MySQLインスタンスの文字セットを問い合わせする
DescribeDBInstanceConfig	MySQLインスタンスのコンフィグレーション情報を問い合わせする
DescribeDBInstanceGTID	MySQLインスタンスのGTIDが有効になっているかを問い合わせする
DescribeDBInstanceRebootTime	MySQLインスタンスの再起動予定時間を問い合わせする

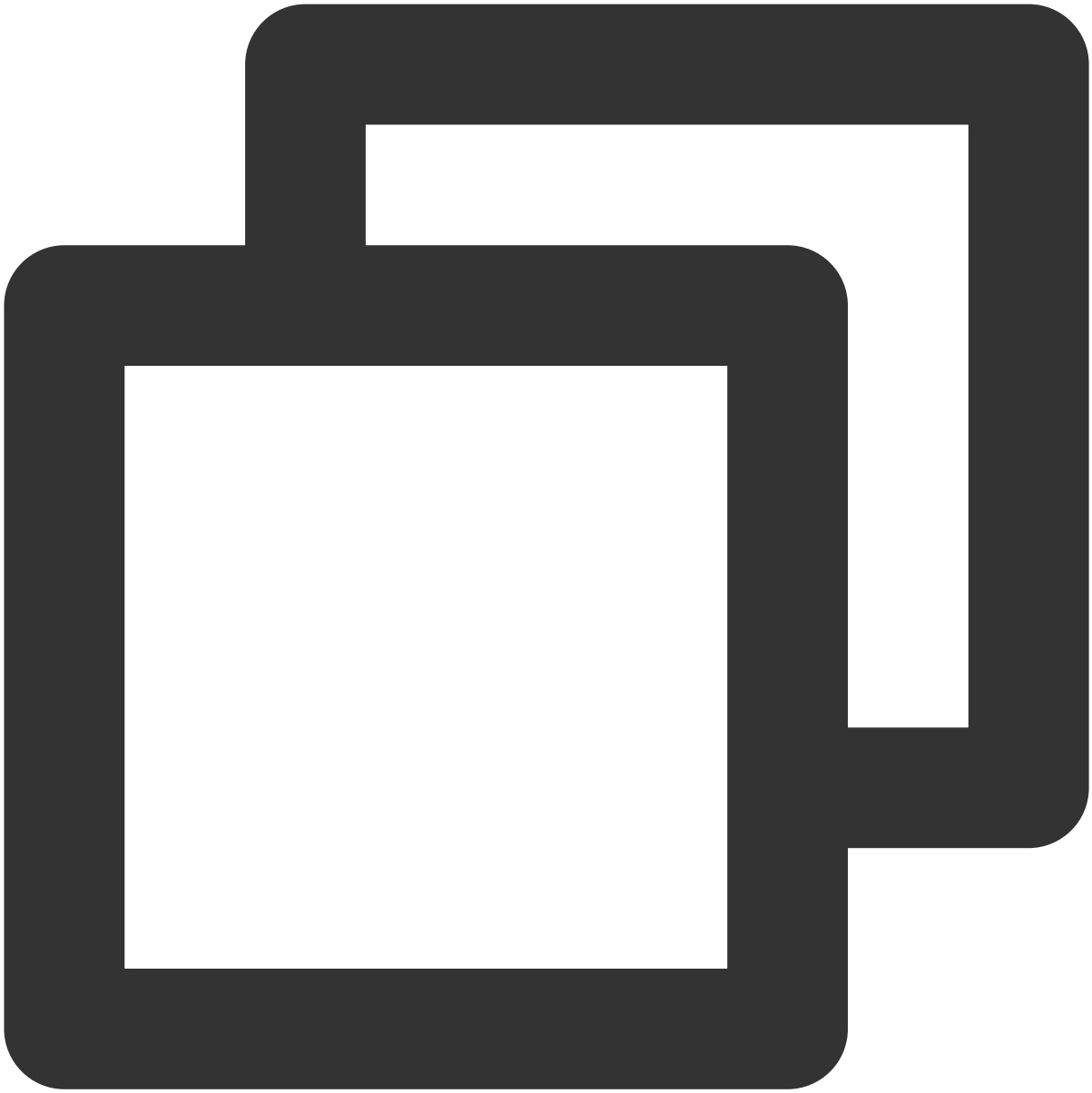
ModifyInstanceParam インスタンスパラメータを変更する|



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

# クラウドAPIエントリモジュールの導入
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudS
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
```

```
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクトをインスタンス化します
    req = models.ModifyInstanceParamRequest()
    req.InstanceIds = ["cdb-1y6g3zj8","cdb-7ghaiocc"]
    req.ParamList = [{"name":"max_connections","currentValue":"100"},{"name":"chara
    #req.ParamList = [{"name":"max_connections","currentValue":"100"}]
    #param = models.Parameter()
    #param.Name = "max_connections"
    #param.CurrentValue = "1000"
    #req.ParamList = [param]


    print req
    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.ModifyInstanceParam(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    msg = traceback.format_exc() # 方式1
    print (msg)
```

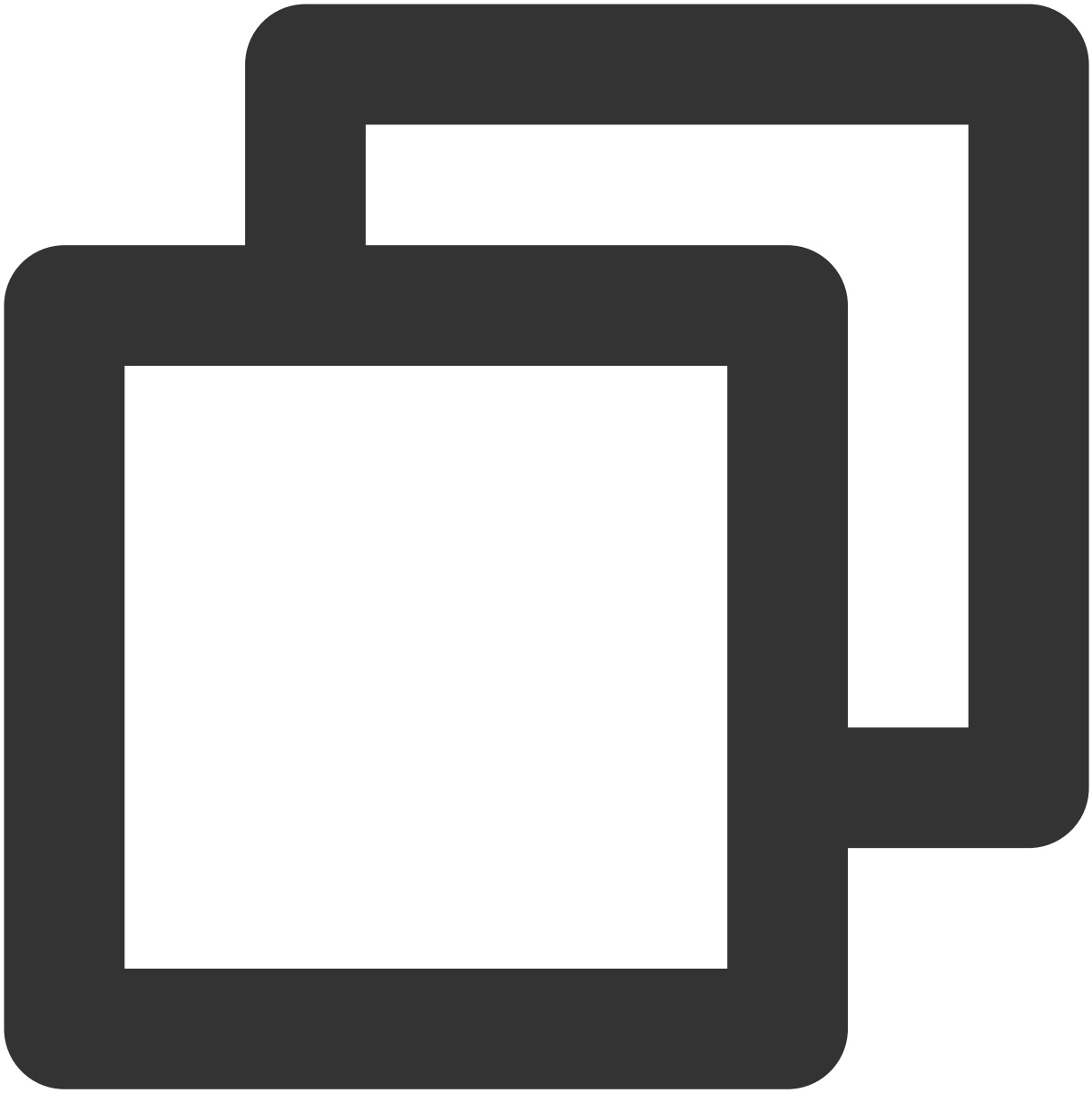
CloseWanService インスタンスのインターネットアクセスを無効にします



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

# クラウドAPIエントリモジュールの導入

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

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.CloseWanServiceRequest()
    req.InstanceId = "cdb-1y6g3zj8"

    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.CloseWanService(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

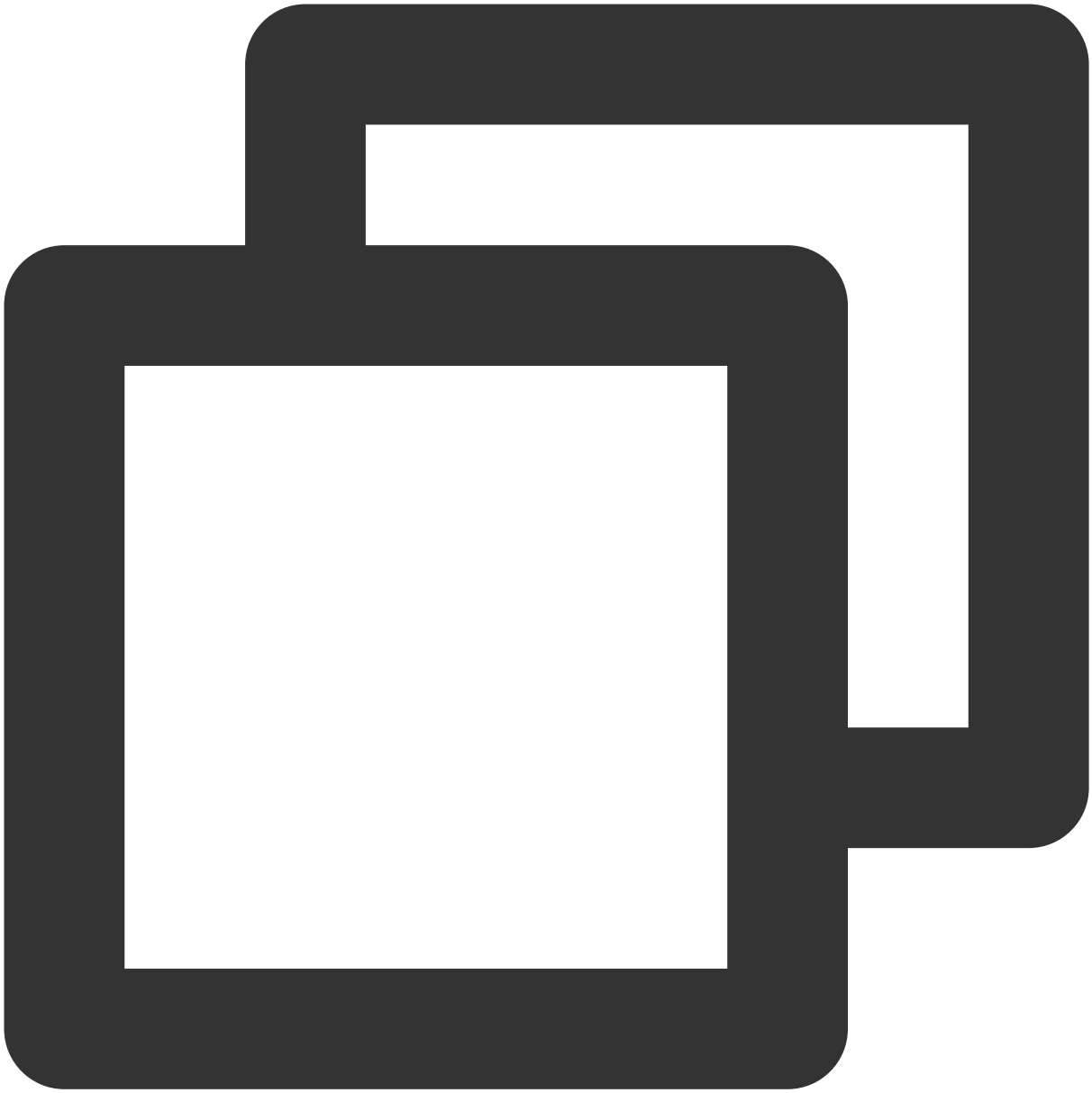
OpenWanService インスタンスのインターネットアクセスを有効にします



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

# クラウドAPIエントリモジュールの導入

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

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.OpenWanServiceRequest()
    req.InstanceId = "cdb-1y6g3zj8"

    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.OpenWanService(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

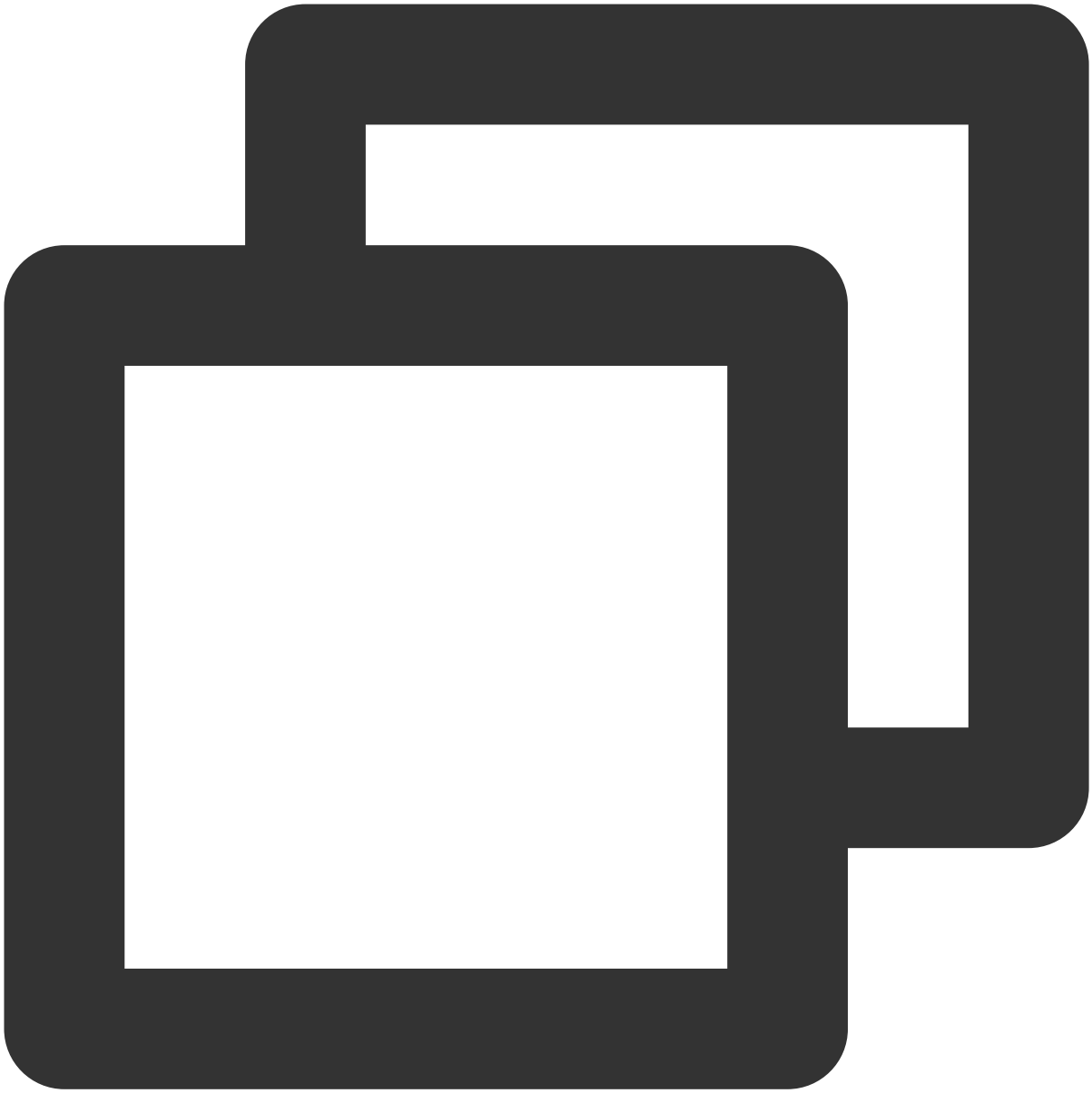
RestartDBInstances　インスタンスを再起動します



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

# クラウドAPIエントリモジュールの導入

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

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.RestartDBInstancesRequest()
    req.InstanceIds = ["cdb-7ghaiocc"]

    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.RestartDBInstances(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

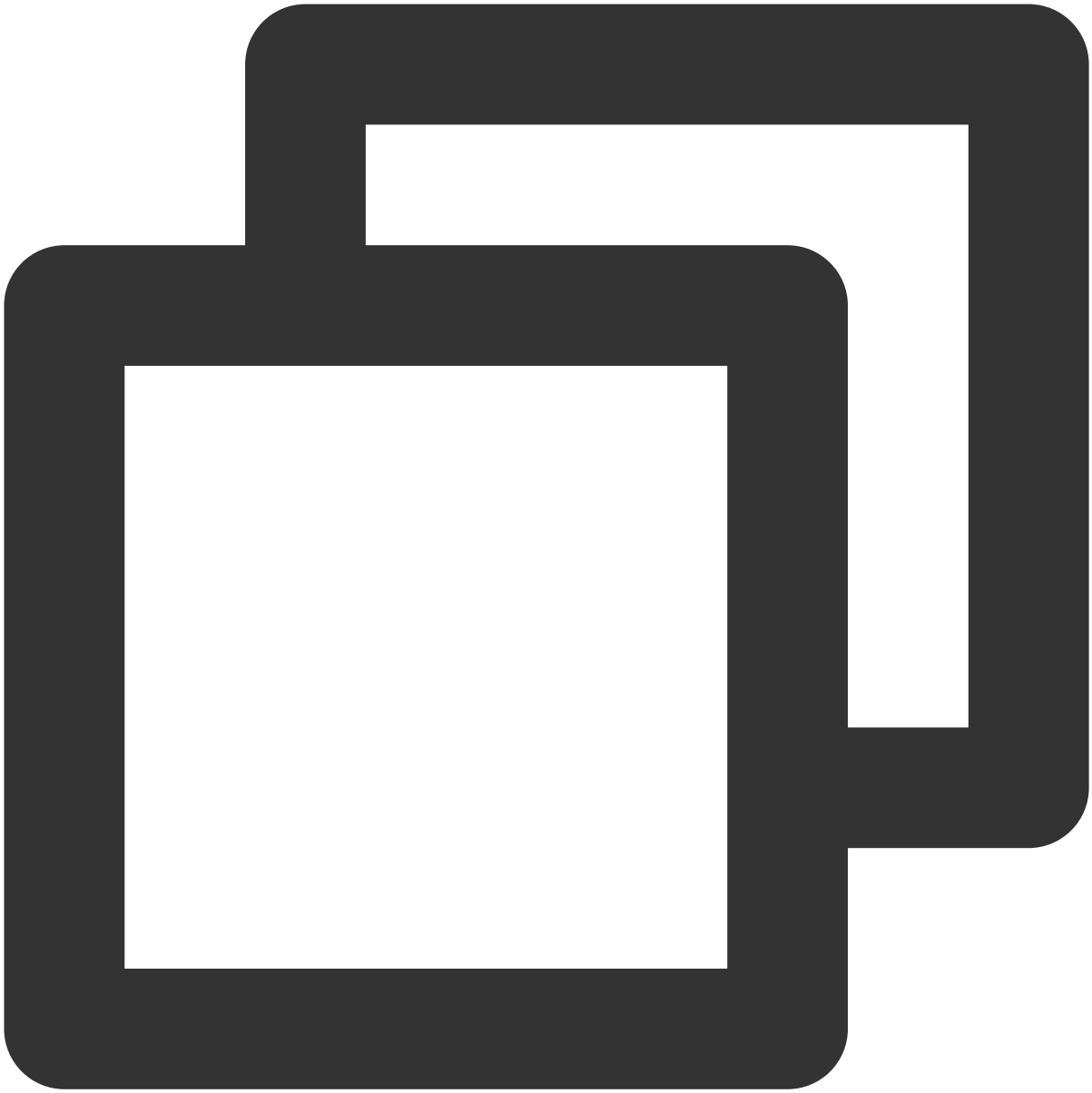
OpenDBInstanceGTID インスタンスのGTIDを有効にします



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

# クラウドAPIエントリモジュールの導入

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

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.OpenDBInstanceGTIDRequest()
    req.InstanceId = "cdb-7ghaiocc"


    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.OpenDBInstanceGTID(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

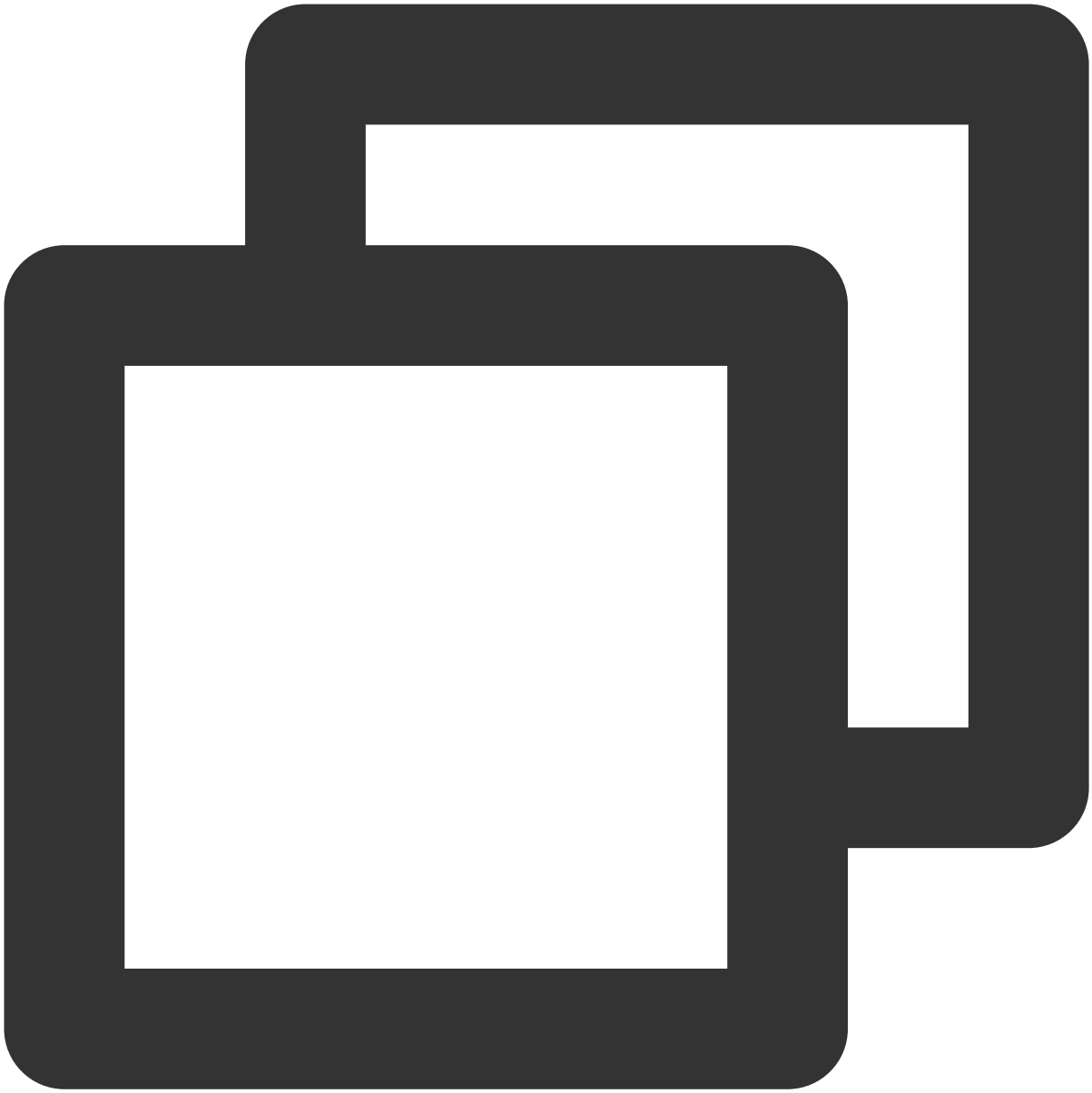
ModifyDBInstanceName　MySQLのインスタンス名を変更します



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

# クラウドAPIエントリモジュールの導入
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudS
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
    cred = credential.Credential("secretId", "secretKey")
```

```
    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-beijing")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.ModifyDBInstanceNameRequest()
    req.InstanceId = "cdb-cukm86n2"
    req.InstanceName = "1s中国語"

    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.ModifyDBInstanceName(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

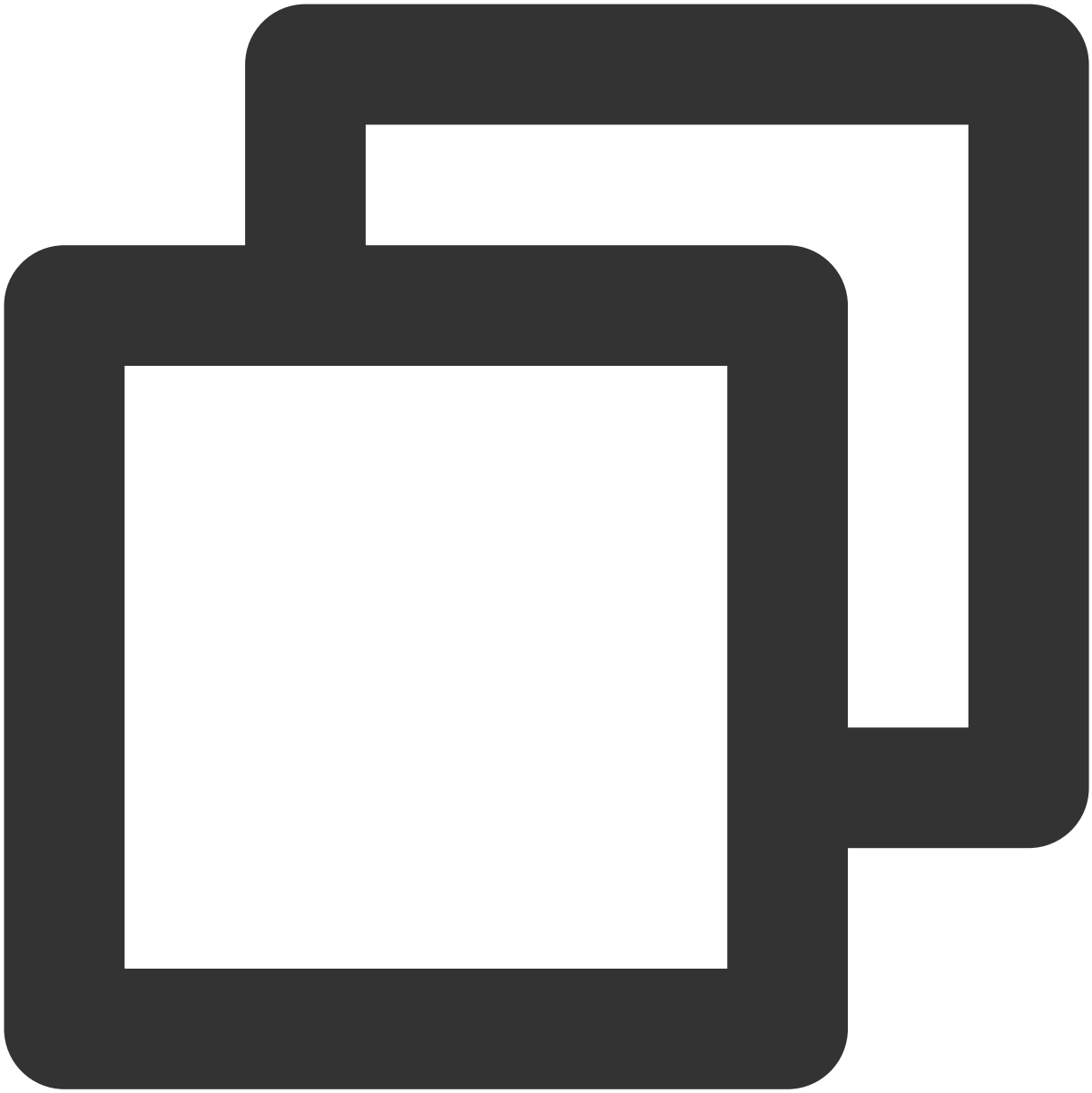
ModifyDBInstanceProject　MySQLインスタンスの所属プロジェクトを変更します



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

# クラウドAPIエントリモジュールの導入
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudS
from tencentcloud.cdb.v20170320 import cdb_client, models
```

```
def DescribeDBInstancesList():
    try:
        # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、
        cred = credential.Credential("secretId", "secretKey")

        #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
        client = cdb_client.CdbClient(cred, "ap-shanghai")

        #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
        req = models.ModifyDBInstanceProjectRequest()
        req.InstanceIds = ["cdb-7ghaiocc"]
        req.NewProjectId =1


        # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクト
        resp = client.ModifyDBInstanceProject(req)

        # json形式の文字列をパッケージに出力します
        print(resp.to_json_string())
    except TencentCloudSDKException as err:
        msg = traceback.format_exc() # 方式1
        print (msg)



DescribeDBInstancesList()
```

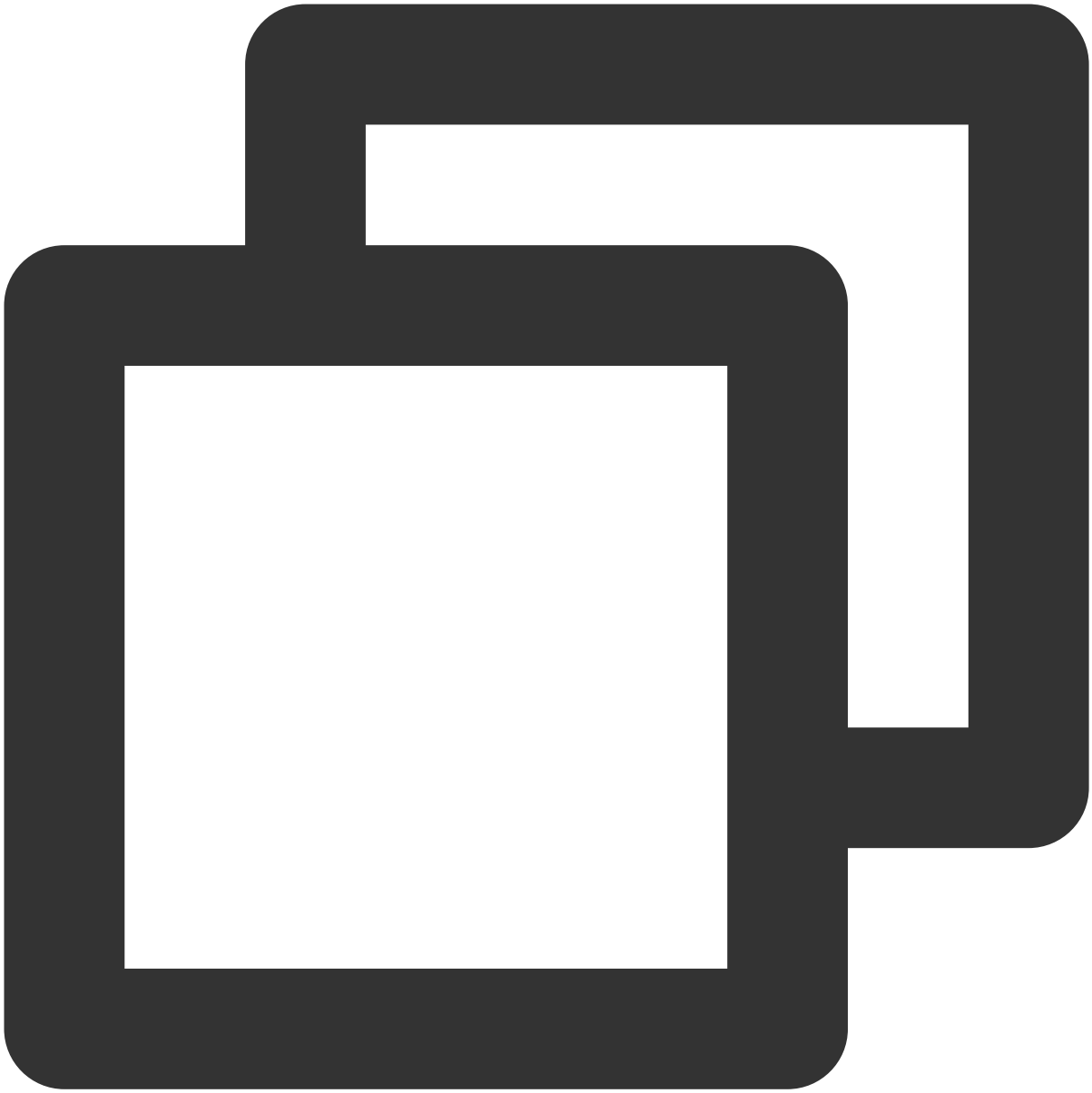
ModifyDBInstanceVipVport MySQLインスタンスのIPとポート番号を変更します



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

# クラウドAPIエントリモジュールの導入
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudS
from tencentcloud.cdb.v20170320 import cdb_client, models
```

```
try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.ModifyDBInstanceVipVportRequest()
    req.InstanceId = "cdb-7ghaiocc"
    req.DstIp = "10.0.0.13"
    req.DstPort =1025
    req.UniqVpcId = 1111

    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.ModifyDBInstanceVipVport(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
        msg = traceback.format_exc() # 方式1
        print (msg)
```

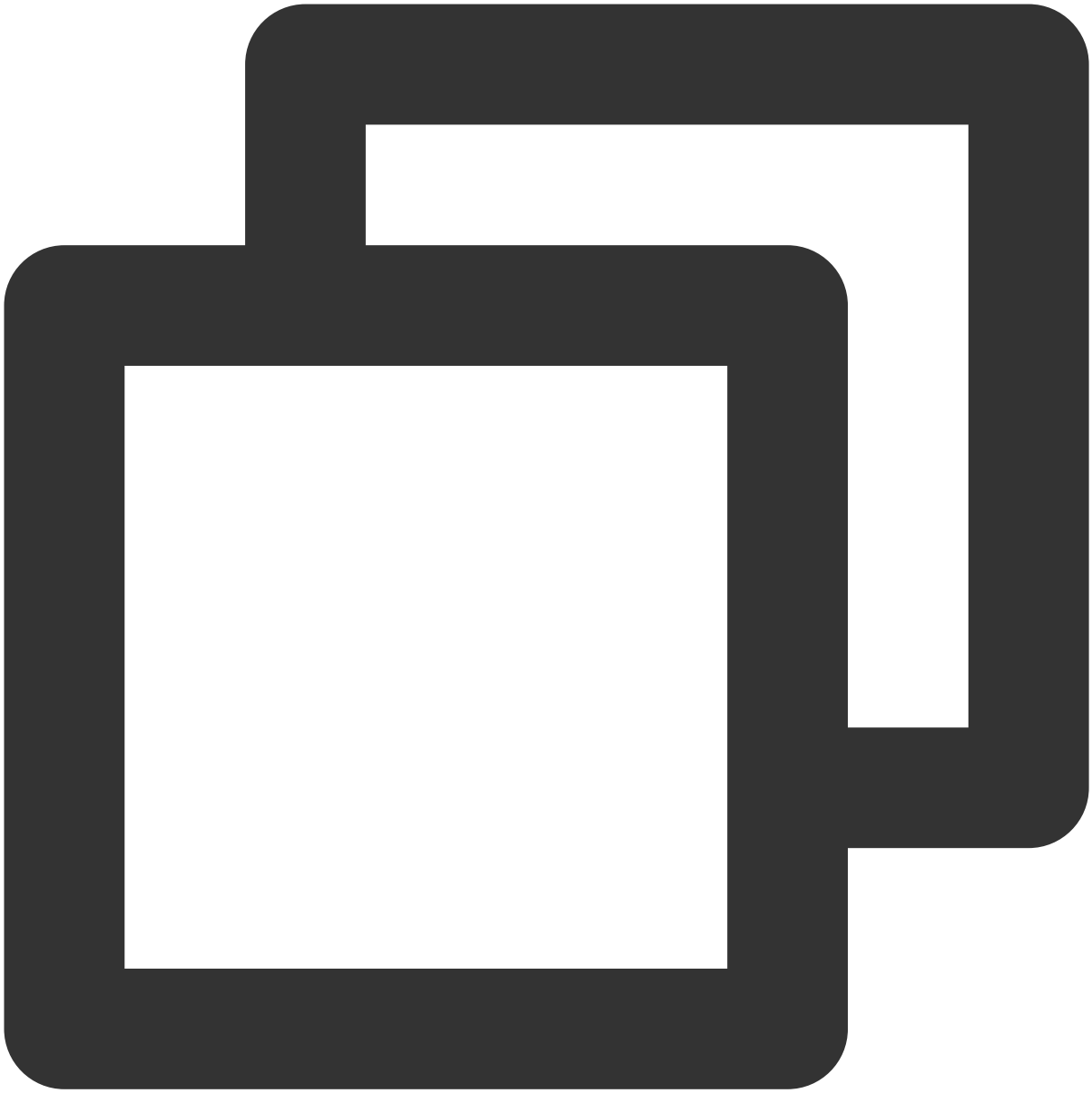
DescribeDBInstanceCharset　MySQLインスタンスの文字セットを問い合わせします



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

# クラウドAPIエントリモジュールの導入

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

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.DescribeDBInstanceCharsetRequest()
    req.InstanceId = "cdb-1y6g3zj8"


    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.DescribeDBInstanceCharset(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

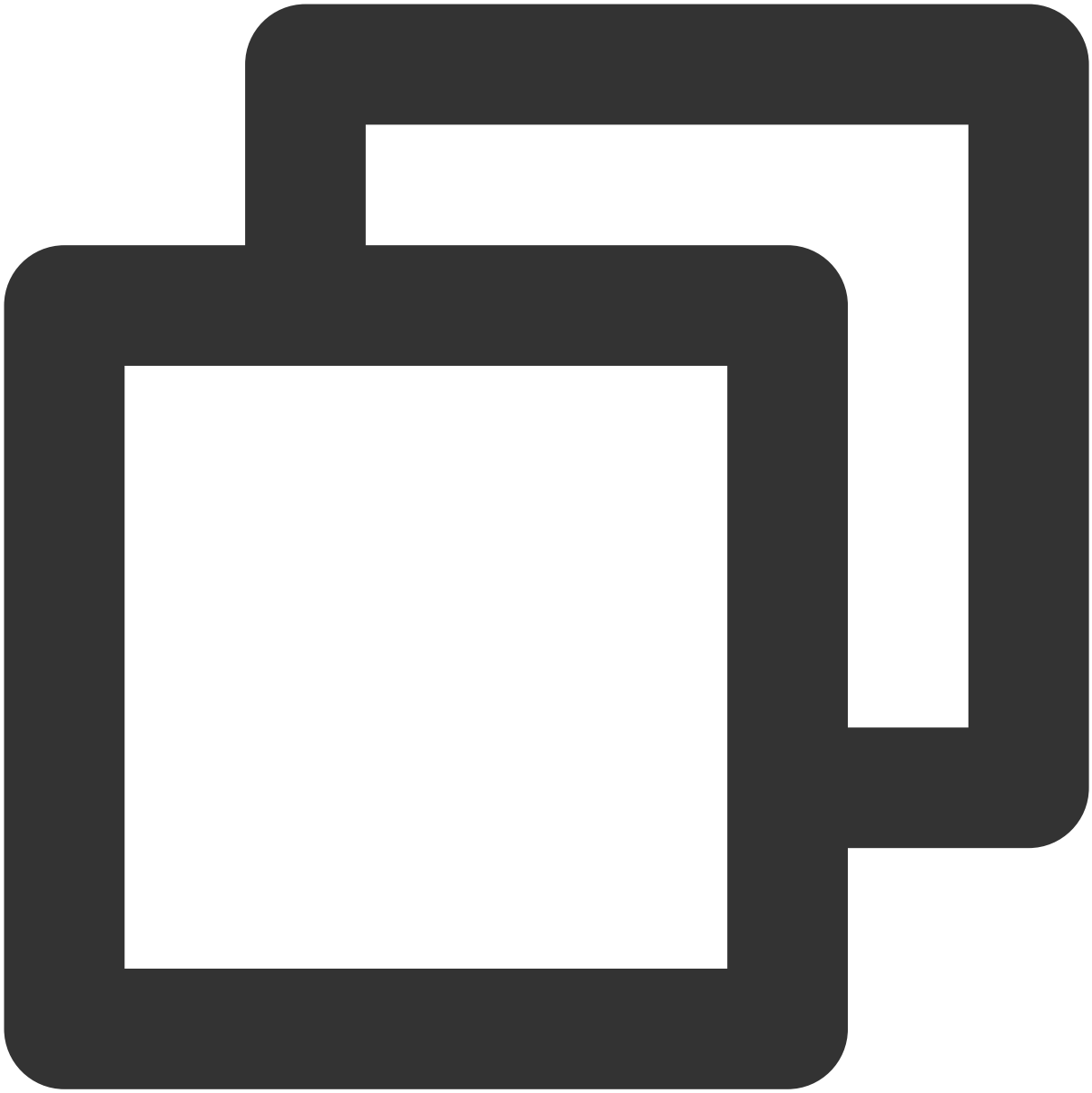
DescribeDBInstanceConfig MySQLインスタンスのコンフィグレーション情報を問い合わせします



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

# クラウドAPIエントリモジュールの導入

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

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.DescribeDBInstanceConfigRequest()
    req.InstanceId = "cdb-1y6g3zj8"


    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.DescribeDBInstanceConfig(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

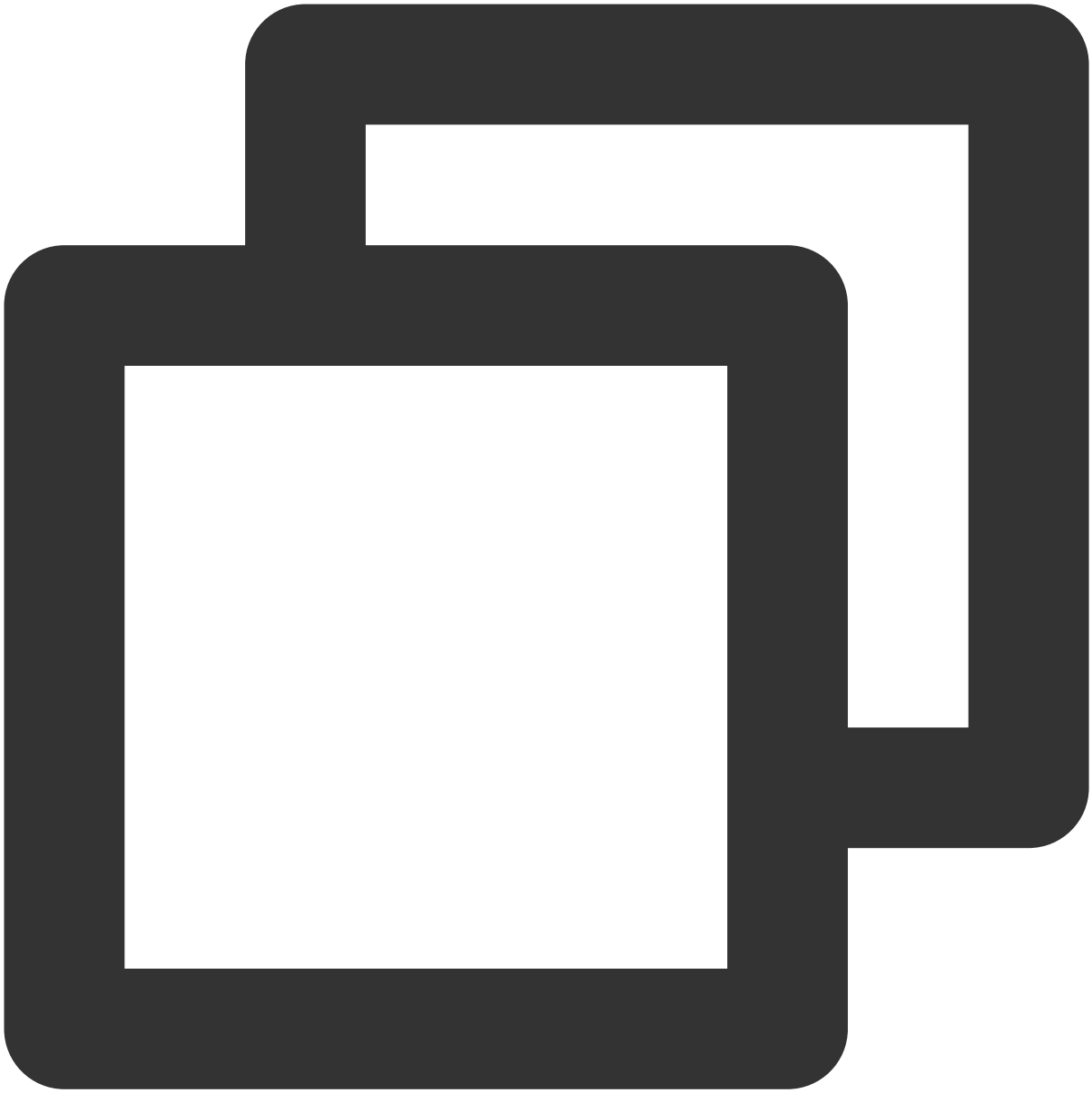
DescribeDBInstanceGTID　MySQLインスタンスのGTIDが有効になっているかを問い合わせします



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

# クラウドAPIエントリモジュールの導入

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

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.DescribeDBInstanceGTIDRequest()
    req.InstanceId = "cdb-1y6g3zj8"


    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.DescribeDBInstanceGTID(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

DescribeDBInstanceRebootTime　　MySQLインスタンスの再起動予定時間を問い合わせします



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

# クラウドAPIエントリモジュールの導入

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

try:
```

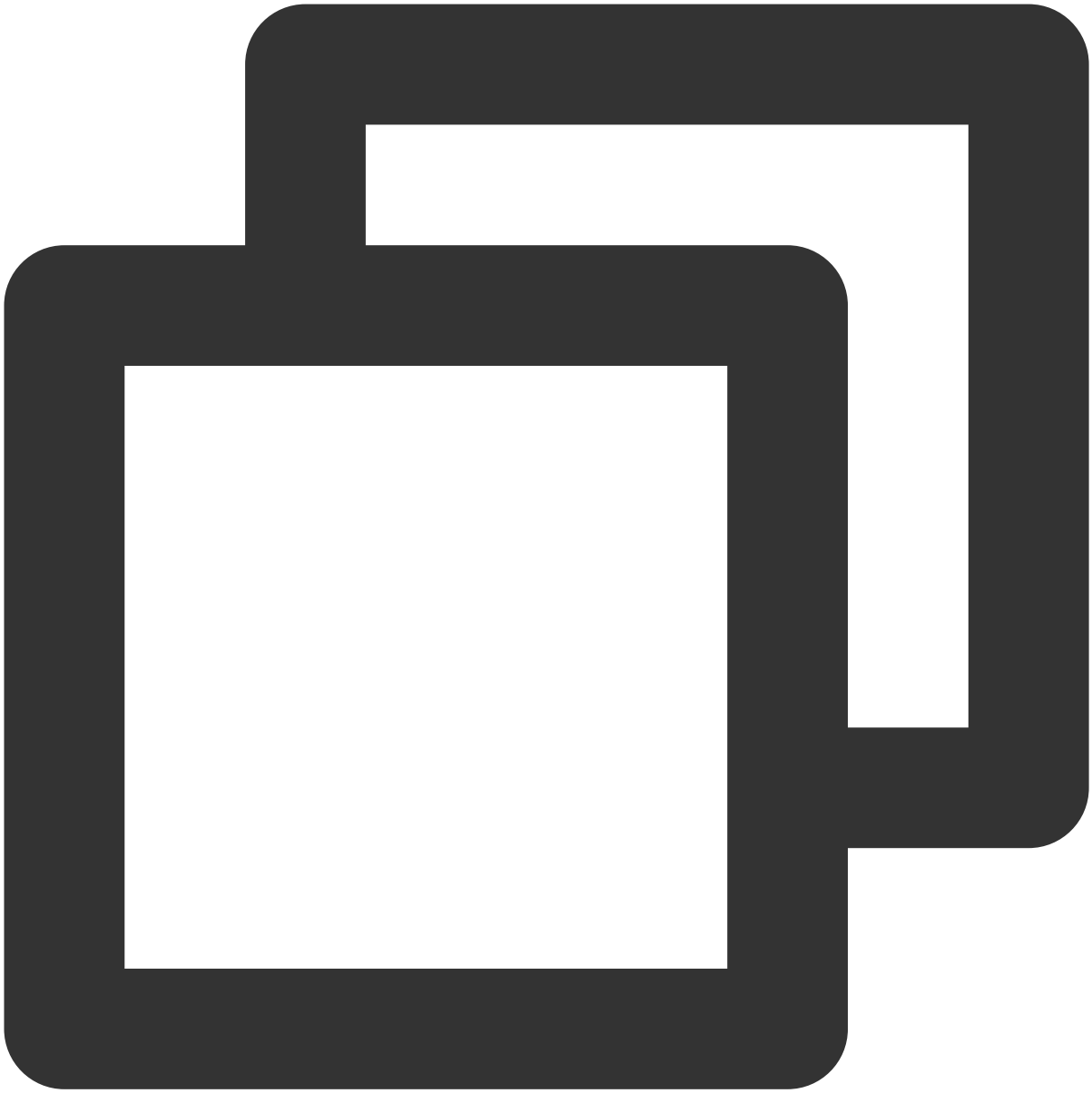
```
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.DescribeDBInstanceRebootTimeRequest()
    req.InstanceIds = ["cdb-1y6g3zj8"]


    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.DescribeDBInstanceRebootTime(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

バックアップタスク

最終更新日：：2024-07-25 17:56:18

API	説明
CreateBackup	MySQLのバックアップを作成する
DeleteBackup	MySQLのバックアップを削除する
DescribeBackupConfig	MySQLのバックアップのコンフィグレーション情報を問い合わせする
DescribeBackupDatabases	バックアップデータベースリストを問い合わせする
DescribeBackupTables	指定したデータベースのバックアップデータテーブルを問い合わせする
DescribeBackups	バックアップログを問い合わせする
DescribeBinlogs	バイナリーログを問い合わせする
DescribeSlowLogs	スロークエリログを問い合わせする
ModifyBackupConfig	データベースのバックアップ構成を変更する

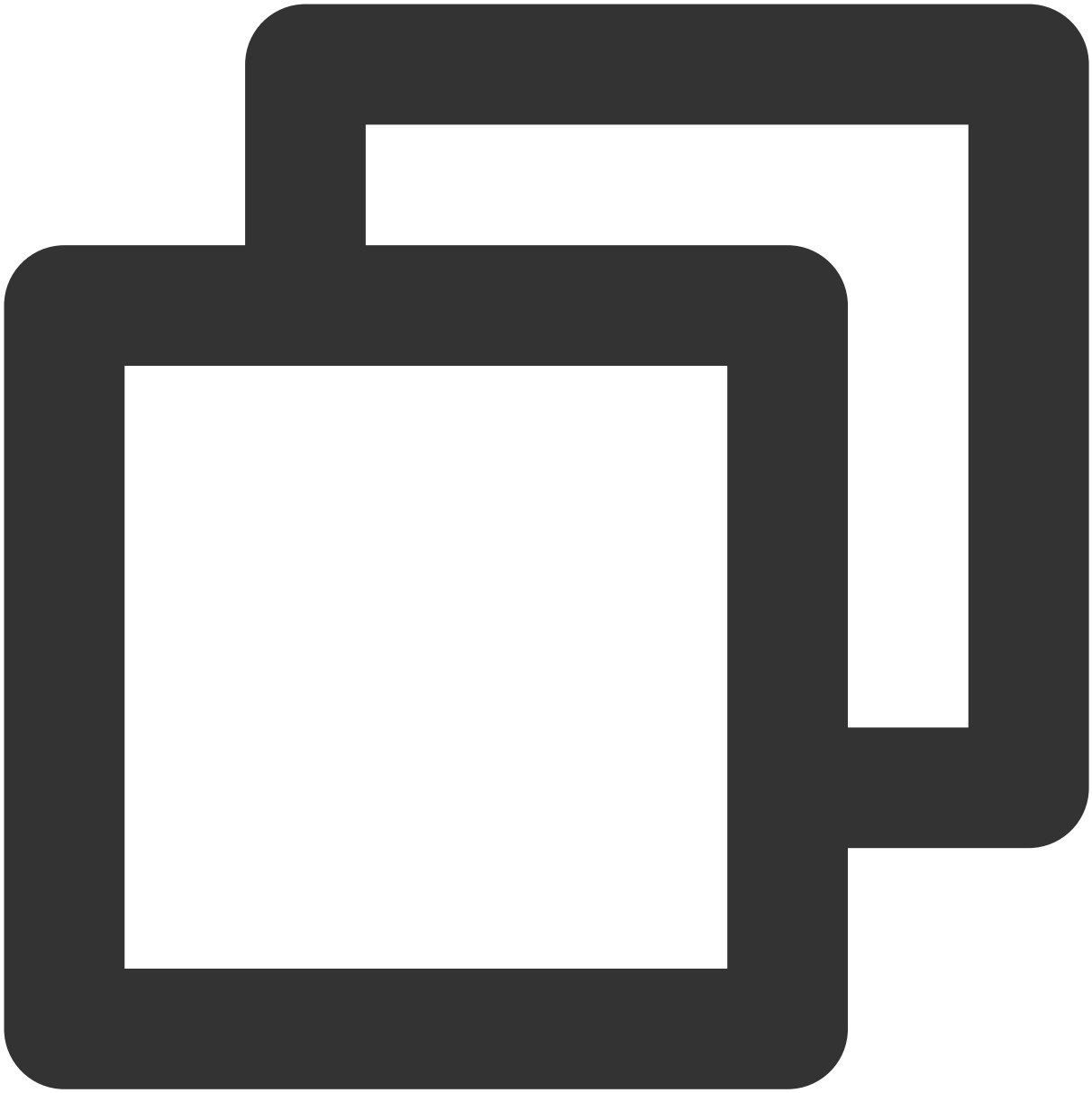
CreateBackup　　MySQLのバックアップを作成します



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

# クラウドAPIエントリモジュールの導入

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

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.CreateBackupRequest()
    req.InstanceId = "cdb-7ghaiocc"
    req.BackupMethod = "logical"

    print req
    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.CreateBackup(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

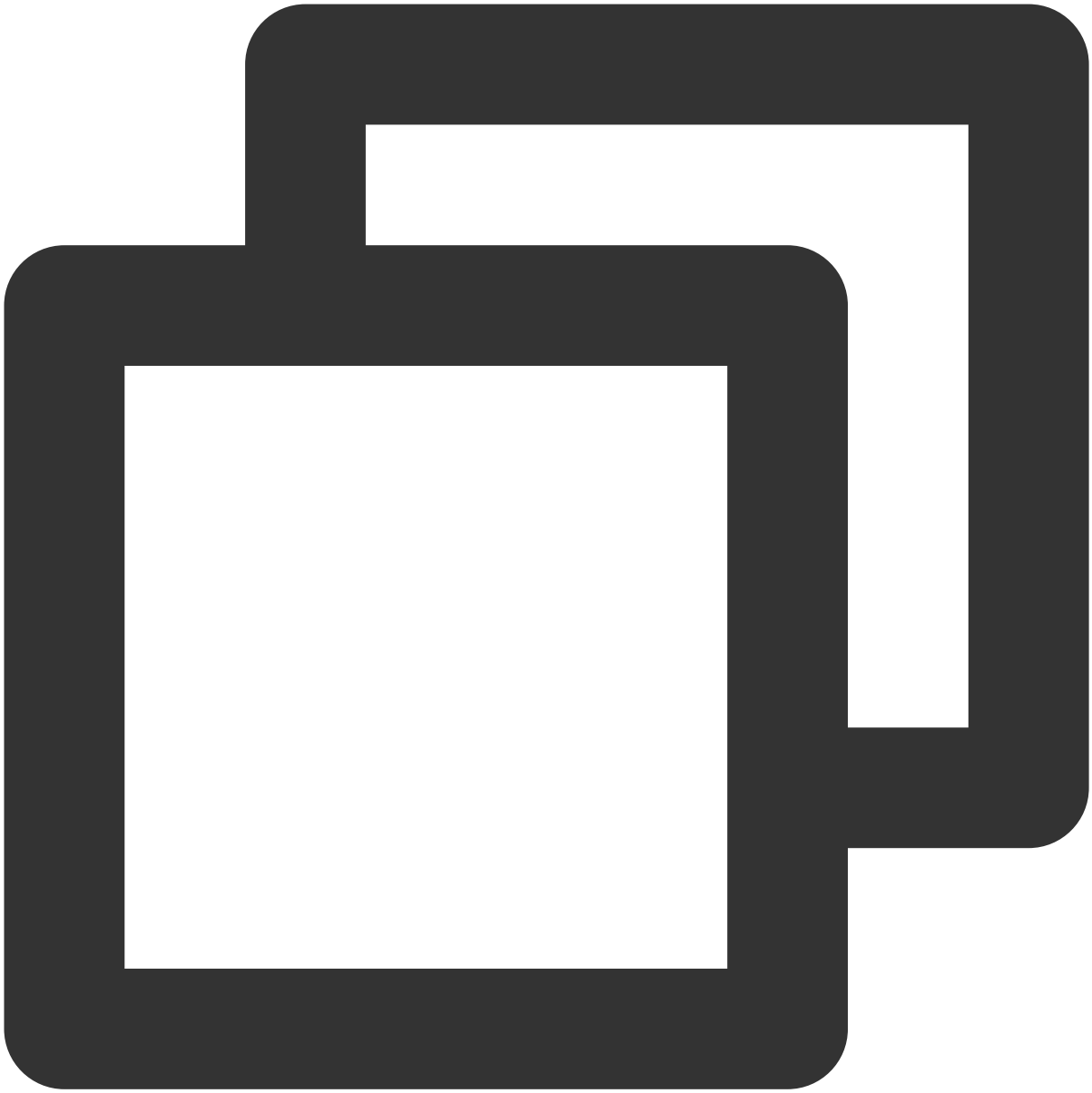
DeleteBackup　MySQLのバックアップを削除します



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

# クラウドAPIエントリモジュールの導入

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

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")



    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.DeleteBackupRequest()

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


    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.DeleteBackup(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

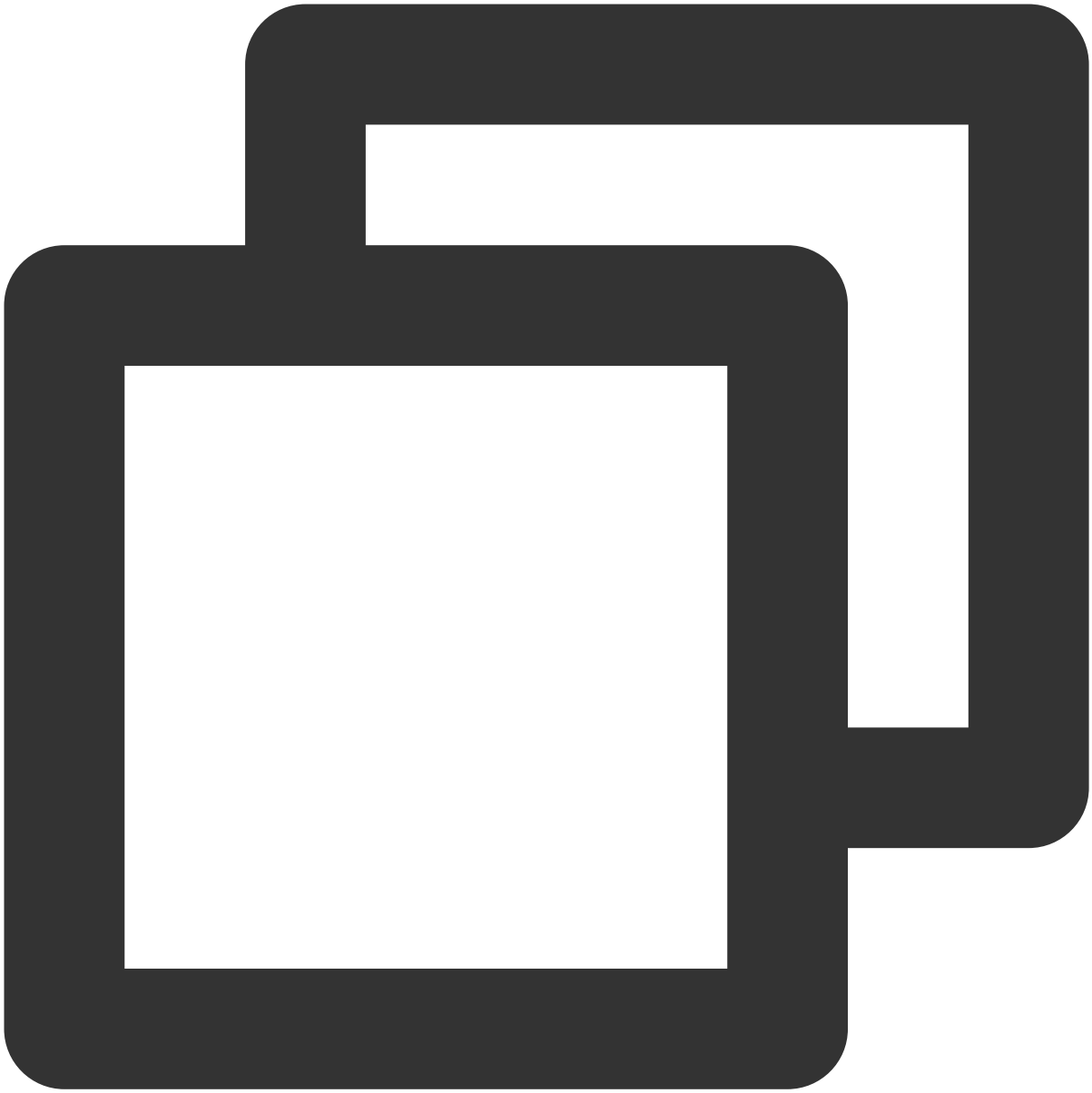
DescribeBackupConfig　　MySQLのバックアップのコンフィグレーション情報を問い合わせします



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

# クラウドAPIエントリモジュールの導入

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

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.DescribeBackupConfigRequest()
    req.InstanceId = "cdb-7ghaiocc"

    print req
    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.DescribeBackupConfig(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

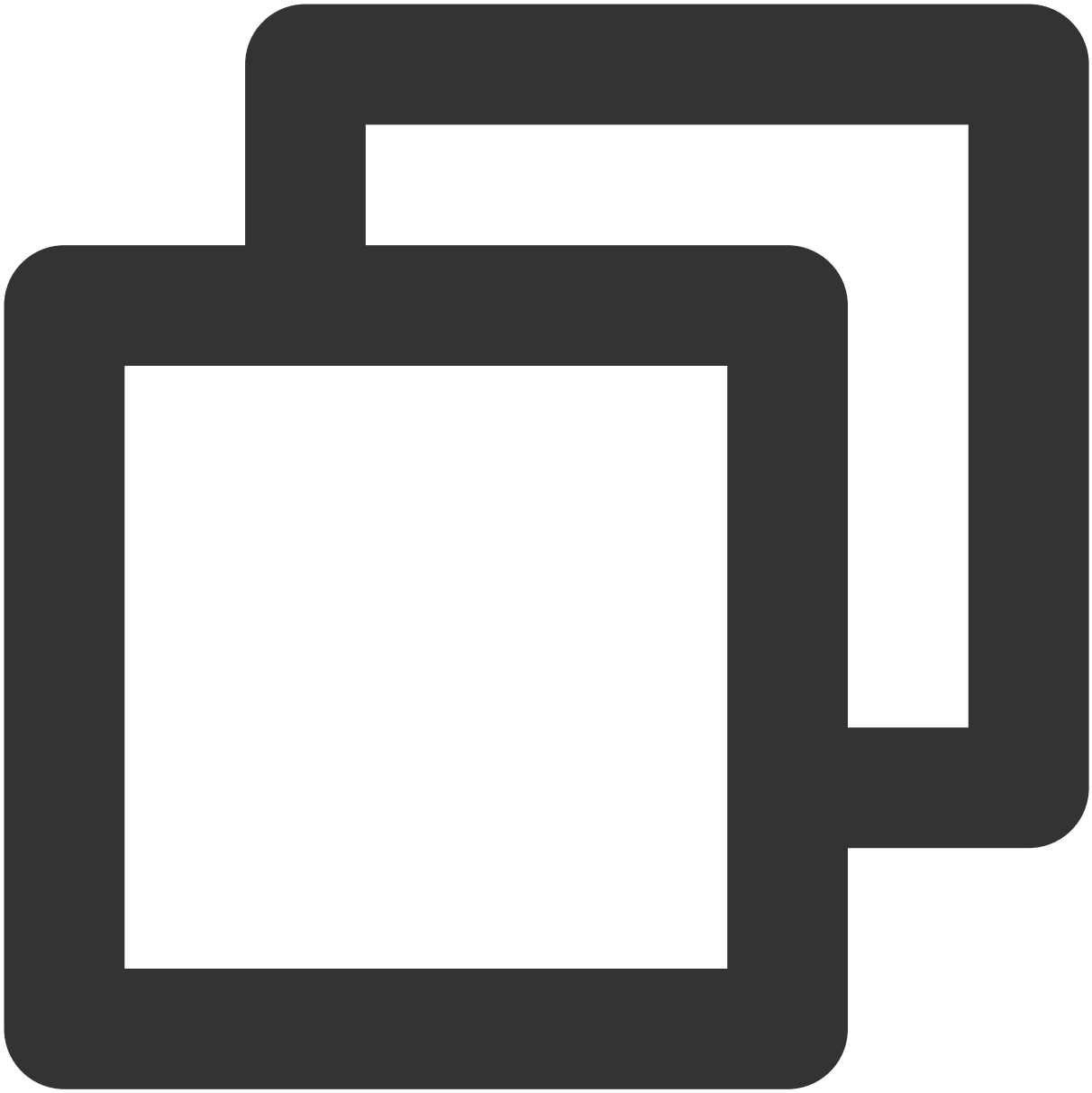
DescribeBackupDatabases バックアップデータベースリストを問い合わせします



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

# クラウドAPIエントリモジュールの導入
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudS
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
```

```
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.DescribeBackupDatabasesRequest()
    req.InstanceId = "cdb-7ghaiocc"
    req.StartTime = "2018-08-02 15:19:19"

    print req
    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.DescribeBackupDatabases(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    msg = traceback.format_exc() # 方式1
    print (msg)
```

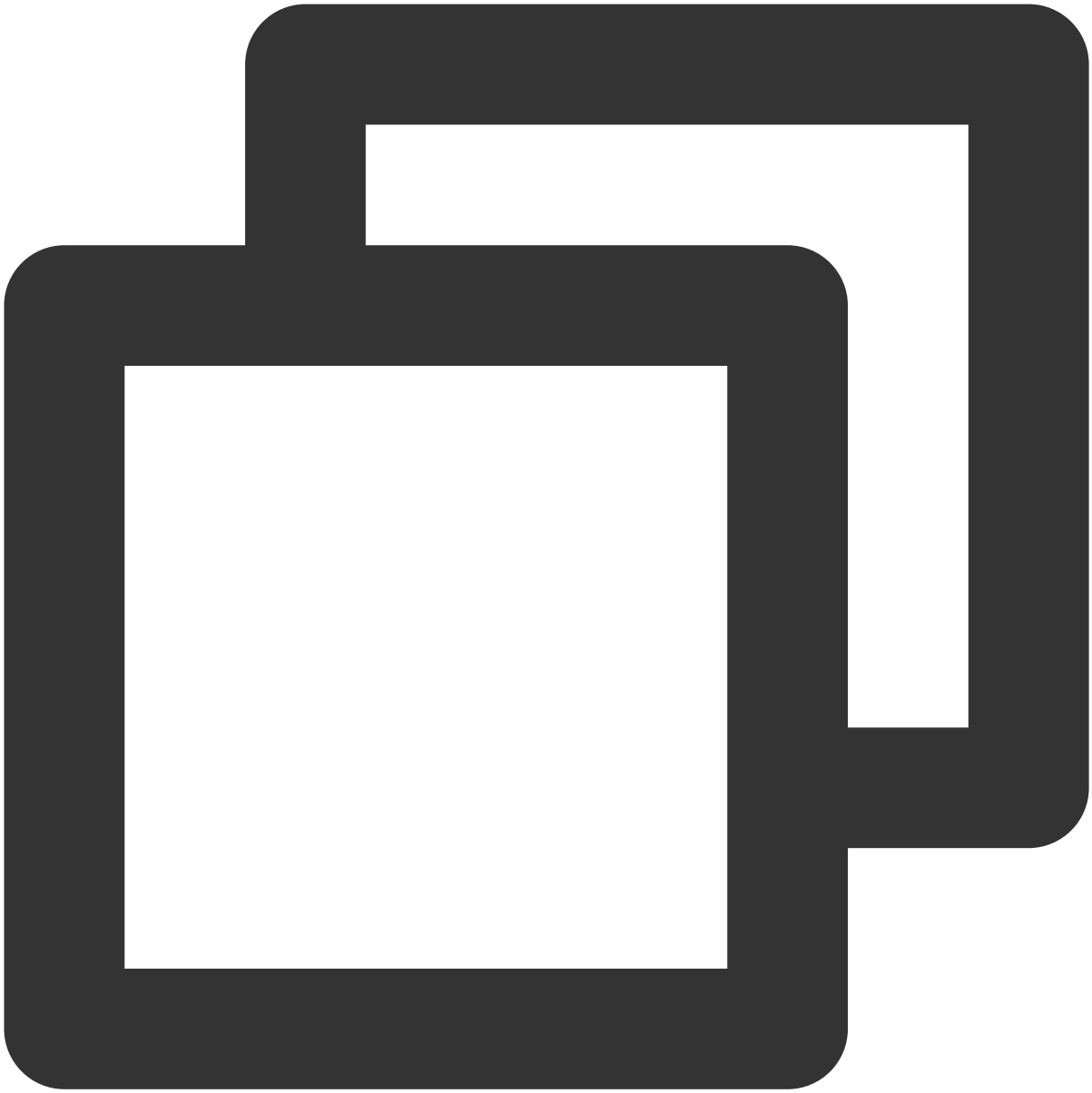
DescribeBackupTables　　指定したデータベースのバックアップデータテーブルを問い合わせします



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

# クラウドAPIエントリモジュールの導入

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

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.DescribeBackupTablesRequest()
    req.InstanceId = "cdb-7ghaiocc"
    req.StartTime = "2018-08-02 15:19:19"
    req.DatabaseName ="sissi"


    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.DescribeBackupTables(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

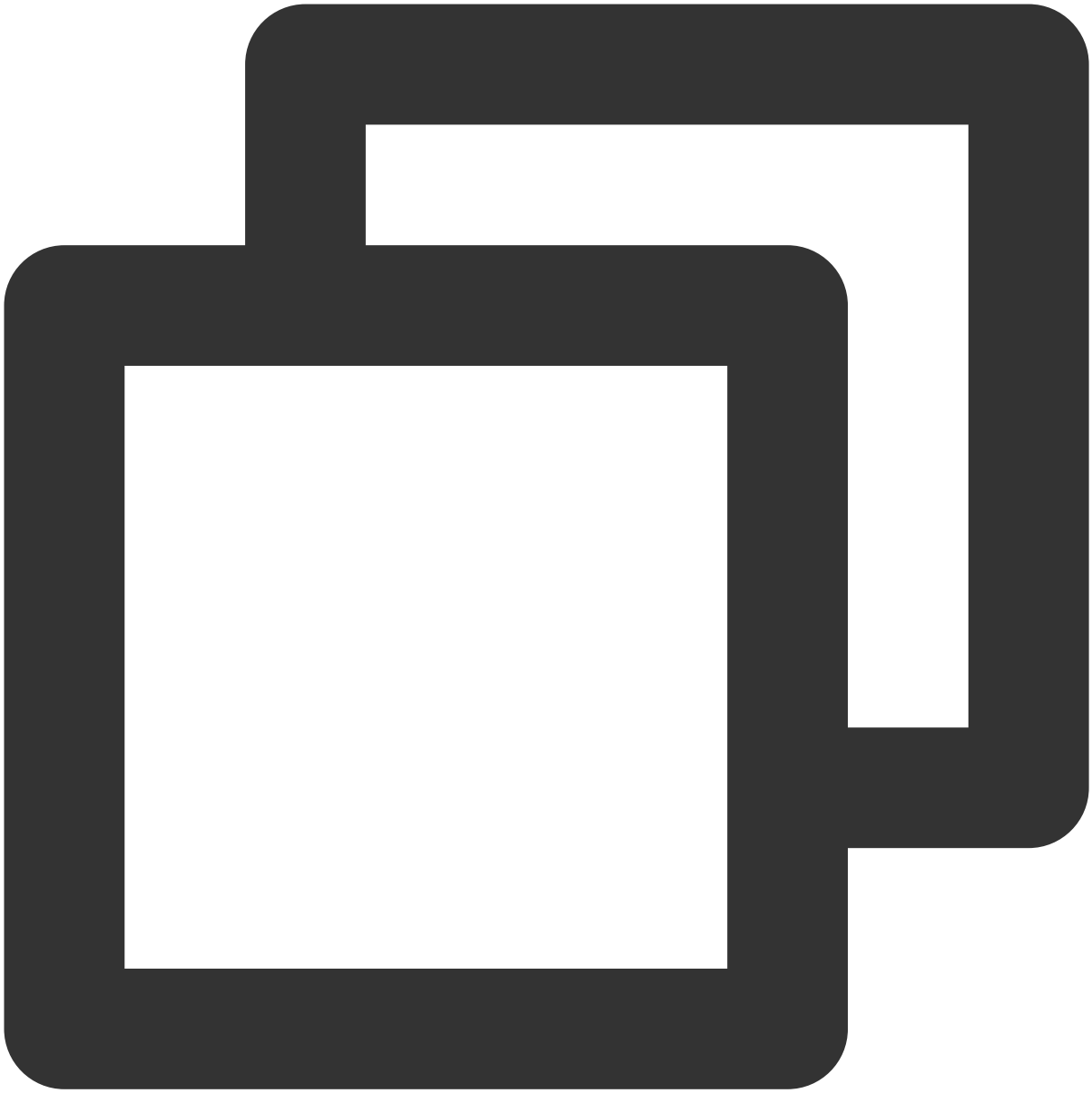
DescribeBackups バックアップログを問い合わせします



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

# クラウドAPIエントリモジュールの導入

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

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.DescribeBackupsRequest()
    req.InstanceId = "cdb-7ghaiocc"



    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.DescribeBackups(req)
    print resp

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

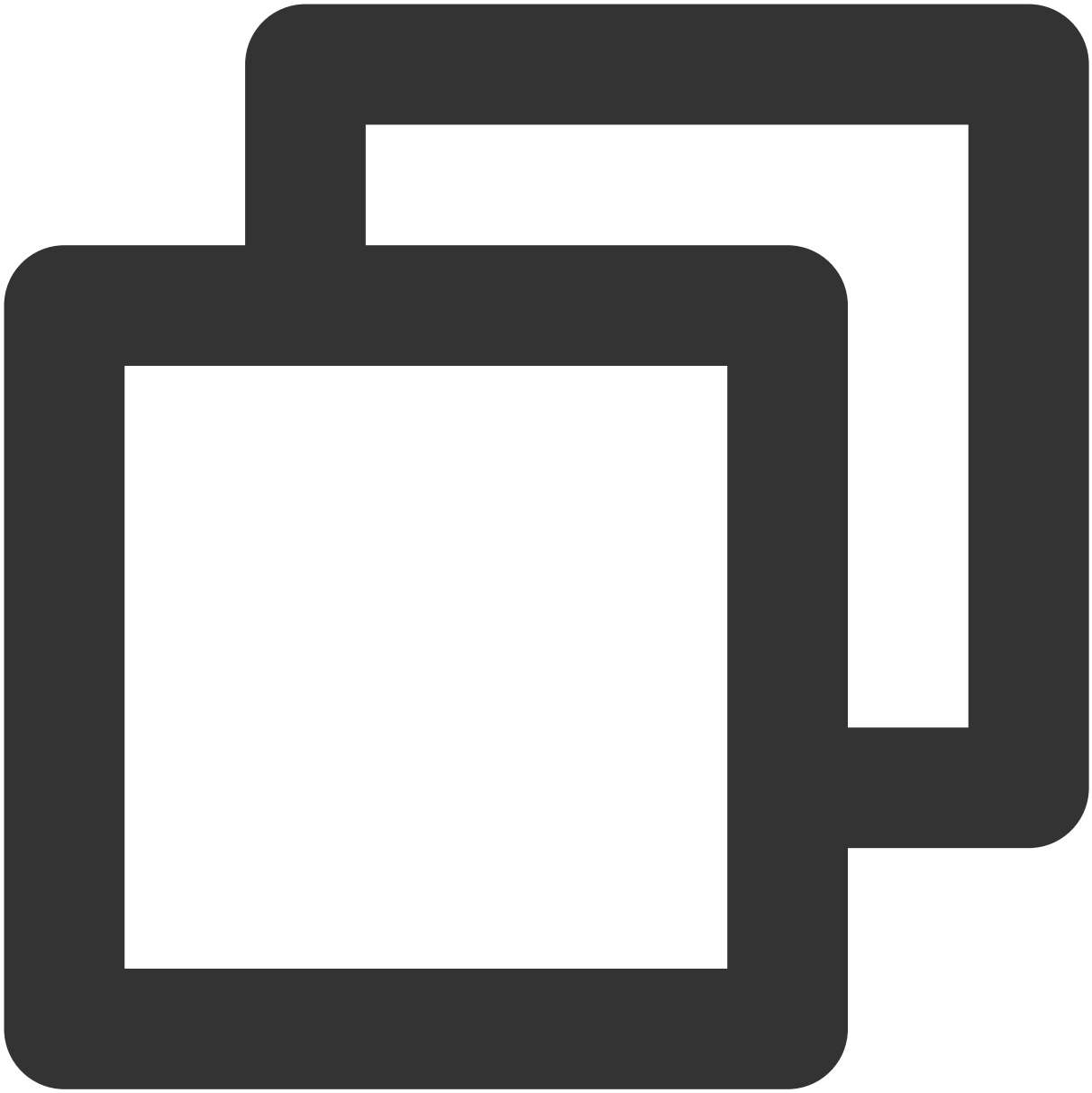
DescribeBinlogs　バイナリーログを問い合わせします



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

# クラウドAPIエントリモジュールの導入

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

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.DescribeBinlogsRequest()
    req.InstanceId = "cdb-7ghaiocc"


    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.DescribeBinlogs(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

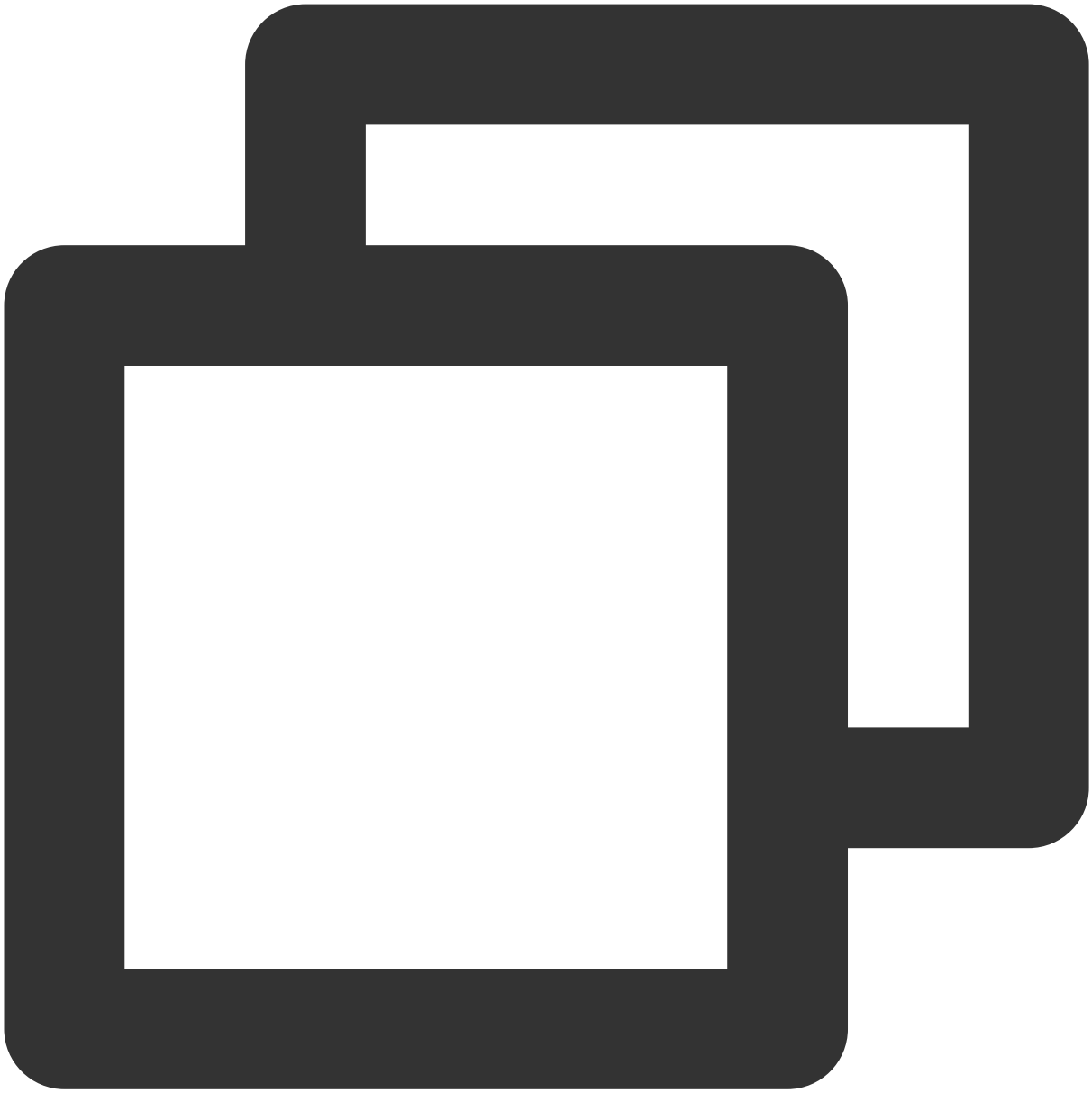
DescribeSlowLogs　スロークエリログを問い合わせします



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

# クラウドAPIエントリモジュールの導入

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

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.DescribeSlowLogsRequest()
    req.InstanceId = "cdb-7ghaiocc"


    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.DescribeSlowLogs(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

ModifyBackupConfig　データベースのバックアップ構成を変更します



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

# クラウドAPIエントリモジュールの導入

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

try:
    # 認証オブジェクトをインスタンス化します。引数としてTencent CloudのアカウントsecretId、sec
```

```
    cred = credential.Credential("secretId", "secretKey")

    #製品（例：cdb）を要求するclientオブジェクトをインスタンス化します
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #要求オブジェクト:req = models.ModifyInstanceParamRequest()をインスタンス化します
    req = models.ModifyBackupConfigRequest()
    req.InstanceId = "cdb-1y6g3zj8"
    req.ExpireDays = 10
    req.StartTime = "06:00-10:00"
    req.BackupMethod = "logical"
    print req


    # clientオブジェクトを通じてアクセスするインターフェースを呼び出すには、要求オブジェクトを渡す
    resp = client.ModifyBackupConfig(req)

    # json形式の文字列をパッケージに出力します
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```