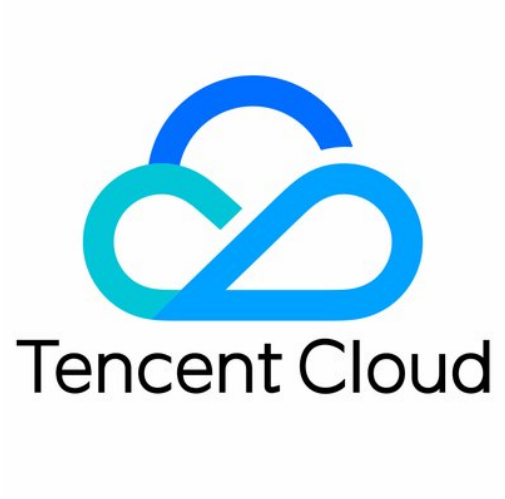
TencentDB for MySQL

사례 튜토리얼

제품 문서



목록:

사례 튜토리얼
TencentDB for MySQL의 사용 규범

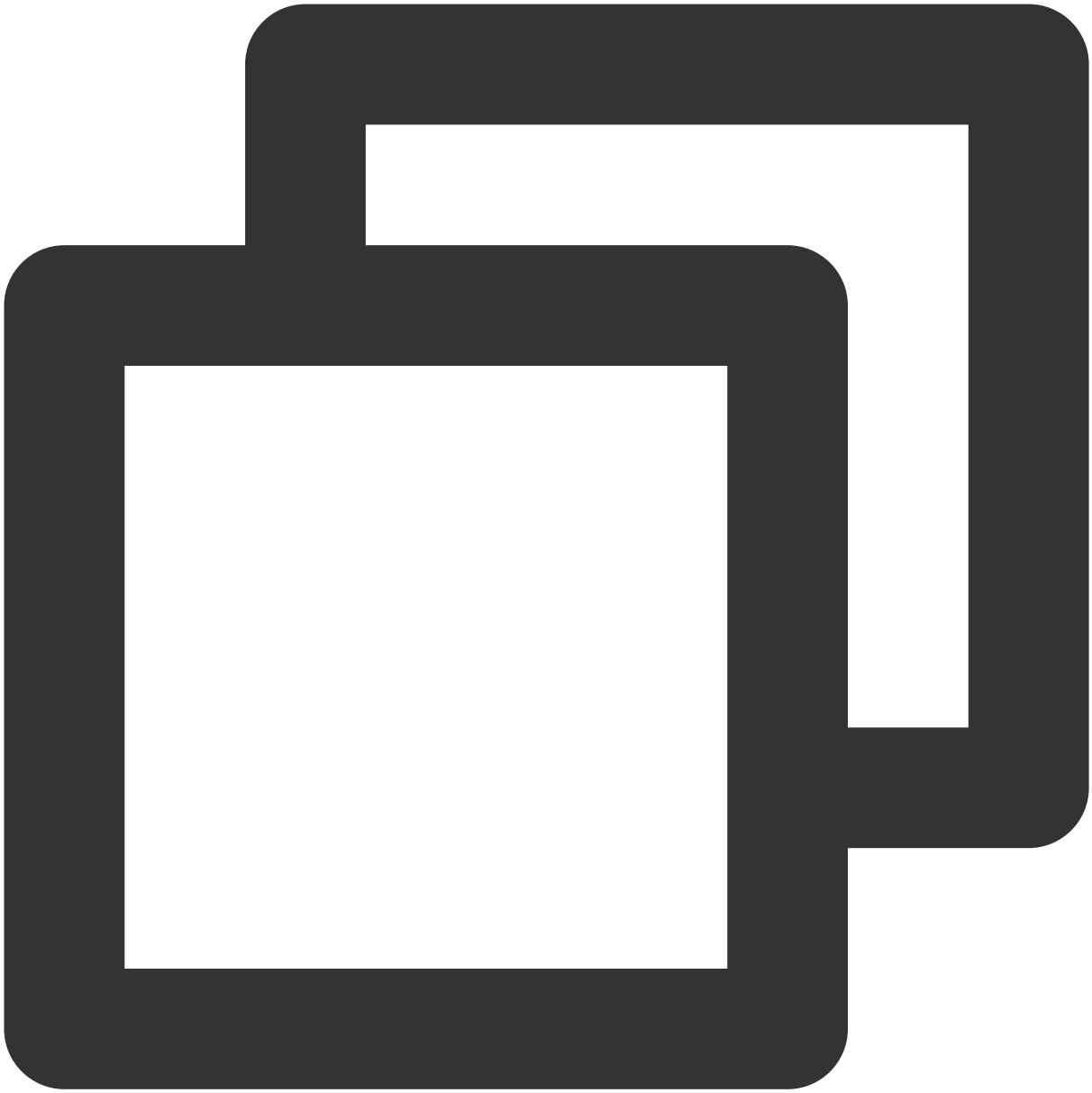
최종 업데이트 날짜: : 2024-07-25 16:38:48

목적

TencentDB for MySQL에 대한 관리 및 점검을 표준화하여 잘못된 작업으로 TencentDB for MySQL에 사용 불가 등의 영향이 발생하지 않도록 합니다.
TencentDB for MySQL가 최적의 성능을 발휘할 수 있도록 데이터베이스 개발자의 효율적인 SQL 작성 지침을 안내합니다.

권한 관리 규범

TencentDB for MySQL의 안정성과 보안성을 위해 TencentDB for MySQL은 super, shutdown, file 권한을 제한합니다. 간혹 TencentDB for MySQL에서 set 명령을 실행할 경우 다음과 같은 오류가 발생합니다.



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

해결 방법: set 관련 매개변수의 수정이 필요한 경우, 콘솔 인스턴스 관리 페이지의 [데이터베이스 관리]>[매개변수 설정] 기능을 사용해 완료할 수 있습니다.
필요에 따라 권한을 부여합니다. 일반적으로 응용 프로그램은 DML(SELECT, UPDATE, INSERT, DELETE) 권한만 부여하면 사용 가능합니다.
객체 권한 최소화가 원칙이며, 일반적인 응용 프로그램 액세스 사용자에게는 데이터베이스 레벨에 따라 권한을 부여합니다.

사용자에게 액세스 권한 부여 시 특정 IP 또는 IP 세그먼트만 액세스를 허용하며, 콘솔에서 보안 그룹을 설정하여 제한할 수 있습니다. 보안 그룹 설정은 반드시 콘솔에서 안내하는 기준에 따라 작업합니다. 공용 네트워크에서 보안 그룹 설정에 액세스하는 시나리오의 경우 반드시 모든 관련 출력 IP를 개방하십시오.
관리 계정은 개발 계정과 분리합니다.

일반적인 작업 규범

주의 사항

데이터베이스 인스턴스 보안성 향상을 위해 보안이 취약한 비밀번호는 사용하지 마십시오.
내부 네트워크 연결 로그인은 클라이언트 측의 클라우드 서버 CVM과 클라우드 데이터베이스 MySQL이 같은 계정과 같은 지역 및 같은 VPC의 기계임을 확보해야 한다.
콘솔에서 다운로드한 binlog 로그를 로컬에서 분석해야 할 경우, 클라이언트 측 MySQL 버전과 TencentDB for MySQL 인스턴스 버전이 반드시 일치해야 합니다. 그렇지 않을 경우 분석 시 글자가 깨지는 현상이 발생할 수 있습니다. 3.4 버전 이상의 mysqlbinlog 사용을 권장합니다.
콘솔에서 내부 네트워크를 통해 CVM에서 콜드 스탠바이 파일을 업로드/다운로드할 때는 url을 따옴표로 묶으십시오. 그렇지 않을 경우 404 오류가 발생할 수 있습니다.

권장 사항

최대한 비즈니스 피크 시간대를 피해 online ddl 작업을 진행하시기 바랍니다. 사용 가능한 툴은 `pt-online-schema-change` 를 참고하십시오.
최대한 비즈니스 피크 시간대를 피해 데이터 일괄 작업을 진행하십시오. 업무가 몰리지 않는 시간대에 나눠서 작업하는 것이 가장 좋습니다.
최대한 한 인스턴스에서 여러 비즈니스를 실행하지 마십시오. 결합도가 지나치게 높으면 비즈니스 간에 서로 영향을 미칠 수 있습니다.
트랜잭션 자동 제출을 비활성화하고 온라인 작업 시 `begin` 을 먼저 실행하면 오작업으로 인한 데이터 손실 위험을 낮출 수 있습니다. 오작업 시 TencentDB for MySQL의 롤백 기능을 사용할 수 있습니다(현재 5일 이내 임의 시간대로 롤백 지원). 관련 테이블이 데이터베이스 간, 테이블 간 로직과 관련되어 있지 않다면 빠른 롤백 또는 고속 롤백을 사용해 데이터를 더욱 빨리 복구할 수 있습니다. 롤백 시 DB 테이블 이름은 '기존 DB 테이블 이름_bak'으로 생성됩니다.
서비스에는 홍보 이벤트 등이 있습니다. 미리 리소스를 예측하고 인스턴스 관련 최적화 작업을 완료하시기 바랍니다. 수요가 많은 경우 해당 서비스 매니저에게 문의하시기 바랍니다.

DB 테이블 설계 규범

주의 사항

TencentDB for MySQL 5.6 이상 버전에서는 MyISAM 엔진과 Memory 엔진을 지원하지 않습니다. Memory 엔진이 필요한 경우 TencentDB for Redis 및 Memcached 사용을 권장합니다. 자체구축 데이터베이스를 TencentDB for MySQL

에 마이그레이션할 경우 자동으로 MyISAM 엔진을 InnoDB 엔진으로 전환합니다.
자동 추가 열이 있는 테이블의 경우, 자동 추가 열에 반드시 1개 이상의 단독 인덱스가 있거나 자동 추가 열로 시작하는 복합 인덱스가 있어야 합니다.
`row_format` 은 반드시 고정되지 않아야 합니다.
모든 테이블에는 반드시 기본 키가 있어야 합니다. 기본 키로 선택할 적합한 열이 없더라도 반드시 무의미한 열을 추가하여 기본 키를 만들어야 합니다. MySQL의 1순위 표준형 기준인 InnoDB 보조 인덱스 리프 노드는 기본 키 값을 저장합니다. 자동으로 추가된 짧은 열을 기본 키로 설정하면 인덱스가 디스크에서 차지하는 용량이 적어져 효율이 향상됩니다. `binlog_format` 이 row인 경우 일괄 삭제 데이터에 기본 키가 없으면 심각한 마스터/슬레이브 딜레이가 발생합니다.
필드를 최대한 NOT NULL로 정의하고 기본값을 작성합니다. NULL은 SQL 개발에 여러 문제를 일으켜 인덱스를 불가능하게 만듭니다. NULL 계산 시 IS NULL과 IS NOT NULL만 사용해 판단합니다.

권장 사항

비즈니스 시나리오 분석과 데이터 액세스(데이터 읽기/쓰기 QPS, TPS, 스토리지 용량 등) 예측을 통해 데이터베이스 사용 리소스를 합리적으로 계획합니다. 콘솔의 클라우드 모니터링 인터페이스에서도 TencentDB for MySQL 인스턴스의 각 항목 모니터링을 설정할 수 있습니다.
데이터베이스 생성 원칙은 동일 유형 비즈니스의 테이블을 같은 데이터베이스에 두는 것입니다. 서로 다른 비즈니스의 테이블은 최대한 같은 데이터베이스를 공유하지 않도록 하고, 프로그램에서 데이터베이스 간의 연결 작업을 실행하지 않도록 합니다. 해당 작업은 이후 빠른 롤백에도 일정한 영향을 미칩니다.
문자 세트는 일괄적으로 utf8mb4를 사용해 글자가 깨지는 리스크를 줄입니다. 일부 복잡한 한자와 이모티콘은 반드시 utf8mb4 방식을 사용해야만 정상적으로 표시됩니다. 문자 세트를 수정하면 수정 후 생성하는 테이블에만 적용되므로, TencentDB for MySQL를 구매하고 인스턴스를 초기화할 때 utf8mb4로 선택하는 것을 권장합니다.
소수 필드는 decimal 유형 사용을 권장합니다. float과 double은 정밀도가 부족합니다. 특히 금전과 관련된 비즈니스는 반드시 decimal을 사용해야 합니다.
대용량 텍스트, 바이너리 데이터, 이미지, 파일 등의 콘텐츠 저장 시 최대한 로컬 디스크 파일로 저장하고, 데이터베이스에서 text/blob을 사용한 저장은 피하십시오. 데이터베이스에는 해당 인덱스 정보만 저장됩니다.
외래 키는 가급적 사용하지 마십시오. 외래 키 로직은 응용 레이어에서 실행하는 것을 권장합니다. 외래 키와 캐스케이드 업데이트는 동시 접속이 높은 시나리오에는 적합하지 않습니다. 삽입 성능을 저하시켜 대규모 동시 접속 시 데드락이 쉽게 발생합니다.
비즈니스 로직과 데이터 스토리지의 결합도를 낮추고, 데이터베이스 스토리지 데이터를 위주로 합니다. 비즈니스 로직은 최대한 응용 레이어를 통해 실행하고, 저장 과정, 트리거, 함수, event, 뷰 등 고급 기능 사용을 최소화합니다. 해당 기능들은 이식성, 확장 가능성이 비교적 낮으므로 인스턴스에 해당 객체가 존재할 경우 기본 값을 definer로 설정하지 않기를 권장합니다. 마이그레이션 계정과 definer의 불일치로 마이그레이션이 실패할 수 있습니다.
단기간 내에 비즈니스가 비교적 큰 규모에 도달하지 못할 경우 파티션 테이블은 사용하지 않는 것을 권장합니다. 파티션 테이블은 주로 보관 관리에 사용되며, 대부분 물류업과 전자 상거래 주문표에 사용됩니다. 비즈니스의 80% 이상이 필드를 구분하여 쿼리하는 경우를 제외하고 성능을 높이는 역할을 하지 않습니다.
읽기 부하가 크고, 일치성 요구가 비교적 낮은(데이터 수신 시 초 단위 딜레이) 비즈니스 시나리오는 읽기 전용 인스턴스를 구매해 슬레이브 데이터베이스로 읽기/쓰기 분리 정책 구현을 권장합니다.

인덱스 설계 규범

주의 사항

업데이트가 매우 빈번하고 구분도가 높지 않은 열에 인덱스를 생성하지 마십시오. 기록 업데이트 시 B+ 트리가 변경됩니다. 업데이트가 잦은 필드에 인덱스를 생성하면 데이터베이스 성능이 크게 저하됩니다.
복합 인덱스 생성 시 구분도가 가장 높은 열을 인덱스 가장 왼쪽에 둡니다. 예를 들어, `select xxx where a = x and b = x;` 에서 a와 b가 함께 그룹 인덱스를 생성하고 a의 구분도가 더 높은 경우 `idx_ab(a,b)'를 생성합니다. 비등호와 등호가 혼합된 판단 조건이 있는 경우 반드시 등호 조건 열을 앞에 놓아야 합니다. 예를 들어, ` where a xxx and b = xxx`는 a의 구분도가 더 높아도 반드시 b를 인덱스의 가장 앞 열에 놓아야 합니다. 인덱스 a에 접근할 수 없기 때문입니다.

권장 사항

단일 테이블의 인덱스 수는 5개를 초과하지 않고, 단일 인덱스에서 필드 수는 5개를 초과하지 않는 것이 좋습니다. 너무 많으면 필터링할 수 없으며, 인덱스도 용량을 차지하여 관리에 리소스가 소모됩니다.
비즈니스 SQL에서 가장 많이 필터링되고, cardinality 값이 비교적 높은 열에 인덱스를 생성하십시오. 비즈니스 SQL에서 접근하지 않는 열에 인덱스를 생성하면 아무 의미가 없습니다. 필드의 고유성이 높을수록, 즉 대표하는 cardinality 값이 높을수록 인덱스 필터 효과도 좋아집니다. 일반적으로 인덱스 열의 cardinality 기록 수가 10% 미만이면 저효율 인덱스라 판단합니다(예: 성별 필터).
varchar 필드에 인덱스를 만들 때 인덱스 길이를 지정하는 것이 좋으며, 직접 전체 열에 인덱스를 생성하는 것은 권장하지 않습니다. 일반적으로 varchar 열은 비교적 길어서 일정 길이를 지정해 인덱스를 만들 경우 이미 구분도가 높으므로, 전체 열에 인덱스를 생성할 필요가 없습니다. 전체 열에 인덱스를 생성하면 용량이 커져 인덱스 점검 비용이 증가할 수 있습니다. count(distinct left(열 이름, 인덱스 길이))/count(*)를 사용해 인덱스 구분도를 확인할 수 있습니다.
인덱스가 중복되는 것을 피하십시오. 두 개의 인덱스 (a,b) (a)가 동시에 존재한다면, (a)는 잉여 인덱스 redundant index에 속합니다. 조회 필터 조건이 a열이면, (a,b) 인덱스로 충분하므로 (a) 인덱스를 단독으로 생성하지 않아도 됩니다.
인덱스 덮어쓰기를 합리적으로 이용하여 IO 부하를 낮춥니다. InnoDB 2단계 인덱스의 리프 노드에는 자체 키 값과 기본 키 값만 저장합니다. SQL 쿼리가 인덱스 열이나 기본 키가 아닌 경우, 이 인덱스를 실행해 먼저 해당하는 기본 키를 찾고, 다시 기본 키에 따라 찾으려는 열을 찾습니다. 이것이 테이블 복구(flashback table)입니다. 이 경우 별도의 IO 부하가 발생하며, 이때 인덱스 덮어쓰기를 이용해 문제를 해결할 수 있습니다. 예를 들어, `select a,b from xxx where a = xxx` 에서 a가 기본 키가 아니라면 a, b 두 개 열의 복합 인덱스를 생성할 수 있습니다. 이 경우 테이블 복구를 하지 않습니다.

SQL 작성 규범

주의 사항

UPDATE, DELETE 작업은 LIMIT를 사용하지 않고, 반드시 WHERE를 사용해 정확히 매칭합니다. LIMIT는 랜덤이므로 해당 작업은 데이터 오류를 일으킬 수 있습니다.
`INSERT INTO t_xxx VALUES(xxx)` 를 사용하지 마십시오. 테이블 구조 변경으로 인한 데이터 오류가 발생하지 않도록, 반드시 삽입하는 열의 속성을 명시적으로 지정해야 합니다.
SQL 명령에서 가장 흔한 인덱스의 효력이 사라지는 상황에 주의해야 합니다.
내장 유형을 전환합니다. 인덱스 a의 유형이 varchar인 경우, SQL 명령을 where a = 1; varchar로 작성하면 int로 전환됩니다.
인덱스 열에 수학 계산과 함수 등의 작업을 진행합니다. 예를 들어, 함수를 사용해 날짜 열을 포맷 처리합니다.
join 열 문자 세트 불일치 시 주의해야 합니다.
여러 열의 정렬 순서 불일치에 주의해야 합니다. 인덱스가 (a,b)인 경우 SQL 명령은 order by a b desclike입니다.
모호한 쿼리를 사용할 경우 문자 세트 유형 `xx%` 형식은 일부 인덱스에 접근할 수 있으나, 다른 상황에서는 인덱스에 접근할 수 없습니다.
네거티브 쿼리(not, !=, not in 등) 사용 시 주의해야 합니다.

권장 사항

필요에 따라 요청하고 `select *` 를 거부하여 다음 문제를 방지합니다.
인덱스 덮어쓰기가 불가능하여 테이블 복구 작업으로 I/O가 증가하는 문제
추가적인 메모리 부담 및 대량의 콜드 데이터를 `innodb_buffer_pool_size` 로 가져와 조회 히트율이 감소하는 문제
추가적인 네트워크 전송 부하
최대한 대규모 트랜잭션 사용을 피하십시오. 대규모 트랜잭션을 소규모 트랜잭션으로 분할하여 마스터/슬레이브 딜레이를 방지합니다.
비즈니스 코드의 트랜잭션을 즉시 제출해 불필요한 락 대기가 발생하지 않도록 합니다.
다중 테이블 join을 적게 사용하고, 큰 테이블은 join을 금지합니다. 두 개의 테이블을 join할 때에는 반드시 작은 테이블을 드라이버 테이블로 만들고, join 열은 문자 세트가 일치해야 하며 모두 인덱스가 있어야 합니다.
LIMIT 페이징을 최적화합니다. LIMIT 80000, 10과 같은 작업은 80010개의 기록을 검색한 후 다시 10개를 반환하는 작업으로, 데이터베이스에 큰 부담을 줍니다. 먼저 첫 번째 기록의 위치를 확인하고 다시 페이징하는 것을 권장합니다. 예시: `SELECT * FROM test WHERE id >= ( SELECT sql_no_cache id FROM test order by id LIMIT 80000,1 ) LIMIT 10 ;`
다중 레이어 서브 쿼리가 중첩된 SQL 명령을 피하십시오. MySQL 5.5 이전의 쿼리 옵터마이저가 in을 exists로 변경하여 인덱스가 유효하지 않게 될 수 있습니다. 외부 테이블이 크면 성능이 저하됩니다.
설명：
위와 같은 상황을 완전히 피하기는 어렵습니다. 이런 조건들을 주요 필터링 조건으로 설정하지 않는 것을 권장합니다. 인덱스의 주요 필터링 조건 뒤에 실행하면 문제가 크지 않습니다.
모니터링에서 전체 테이블 스캔 양이 비교적 많은 것을 발견할 경우, 콘솔에서 매개변수를 `log_queries_not_using_indexes` 로 설정하고, 잠시 후 슬로우 로그 파일 분석을 다운로드할 수 있습니다. 단, 너무 오래 켜두면 슬로우 로그가 폭증할 수 있습니다.

비즈니스 런칭 전 필요한 SQL 심사를 점검해야 하며, 상시 유지보수 시 주기적으로 슬로우 쿼리 로그를 다운로드하여 맞춤형으로 최적화해야 합니다.

애플리케이션 구성 자동 재연결

최종 업데이트 날짜: : 2024-07-25 16:38:48

본 문서는 인스턴스 변경 시 몇 초간 연결이 끊기면서 나타나는 영향 및 자동 재연결 기능 설정을 소개합니다.

상황

TencentDB for MySQL에서 데이터베이스 인스턴스 사양 조정 및 데이터베이스 엔진 버전 업그레이드 등의 작업을 진행하거나, 마스터 인스턴스 부하가 높아져 hang 되고, 하드웨어 장애가 발생 시, 인스턴스를 변경해야 할 수 있습니다. 인스턴스 변경 시 몇 초간 연결이 끊길 수 있습니다.
만약 응용 프로그램에 자동 재연결 기능이 비활성화 되어 있을 경우, 마스터/슬레이브 변경 후 애플리케이션 연결에 오류가 발생해 서비스의 정상적인 액세스에 영향을 미칠 수 있습니다.
응용 프로그램에 자동 재연결 기능을 적용하고, 인스턴스를 선택해 점검 시간 내에 변경하시기를 권장합니다.

자동 재연결 기능 설정

마스터/슬레이브 변경으로 인한 애플리케이션 연결 오류를 방지하기 위해 TencentDB for MySQL의 응용 프로그램에 자동 재연결 기능을 적용하고, 연결 풀에 connectTimeOut과 socketTimeOut 매개변수를 설정하시기를 권장합니다. 서비스 시나리오에 따라 적절한 매개변수 값을 설정하십시오. OLTP(On-Line Transaction Processing)의 서비스 시나리오에 따라 20초로 일괄 설정하시기를 권장합니다.
설명：
connectTimeOut：응용 프로그램과 데이터베이스 서버 TCP 연결 타임 아웃 시간은 적어도 응용 프로그램에서 데이터베이스 서버까지의 응답 시간보다 크도록 권장합니다.
socketTimeOut：TCP 연결을 통해 데이터 패키지를 발송한 후 응답 대기 초과 시간을 단일 SQL 최대 실행 시간으로 설정하시기를 권장합니다.

MySQL 마스터 인스턴스 매개변수 수정의 영향

최종 업데이트 날짜: : 2024-07-25 16:38:48

TencentDB for MySQL 인스턴스에 대해 사용자는 콘솔에서 마스터 인스턴스의 매개변수를 수정할 수 있습니다. 이 중 일부 중요 매개변수는 부적절한 수정 방식을 사용할 경우 재해 복구 비정상 또는 데이터 불일치를 초래할 수 있습니다. 본 문서에서는 아래와 같이 매개변수 수정 후의 영향을 소개합니다.

lower_case_table_names

기본값: 0
**용도: **데이터베이스 및 테이블을 생성할 때, 스토리지 및 쿼리 시 대소문자에 민감한지 여부. 해당 매개변수의 설정 값은 0, 1이며 기본값은 0입니다. 0은 데이터베이스 및 테이블 생성 시 스토리지 및 쿼리 모두 대소문자를 구분함을 의미하며, 1은 그 반대를 의미합니다.
**영향: **마스터 인스턴스 수정 후, 재해 복구의 매개변수를 동기화 수정할 수 없습니다. 마스터 인스턴스 대소문자에는 민감하고, 재해 복구 대소문자에는 민감하지 않을 경우(예: 마스터 인스턴스에서 테이블을 2개 생성하고, 테이블 이름이 각각 Test, TEst), 재해 복구가 애플리케이션 대응 로그에 있으면 데이터 동기화 상태 오류가 발생할 수 있습니다. 오류의 원인은 TEst 테이블 이름이 이미 존재하기 때문입니다.

auto_increment_increment

기본값: 1
용도: 자동 증가 AUTO_INCREMENT의 증가량 값에 사용. 해당 매개변수의 설정 가능 범위는 1~65535이며, 기본값은 1입니다.
영향: 마스터 인스턴스에서 매개변수를 수정하면 재해 복구 인스턴스의 매개변수를 동기화 수정할 수 없습니다. binlog_format을 statement로 설정 시, 실행 명령어만 기록됩니다. 이때 마스터 인스턴스에서 자동 증가된 증가량 값은 수정하나, 재해 복구 인스턴스는 동기화 변경하지 않아 마스터/슬레이브의 데이터 불일치가 발생할 수 있습니다.

auto_increment_offset

기본값: 1
용도: 자동 증가 AUTO_INCREMENT의 시작값(오프셋)에 사용. 해당 매개변수의 설정 가능 범위는 1~65535이며, 기본값은 1입니다.
영향: 마스터 인스턴스에서 매개변수를 수정하면 재해 복구 인스턴스의 매개변수를 동기화 수정할 수 없습니다. 마스터/슬레이브에서 Auto Increment의 시작값은 수정하나, 재해 복구 인스턴스는 동기화 변경하지 않아 마스터/슬레이브의 데이터 불일치가 발생할 수 있습니다.

sql_mode

기본값: NO_ENGINE_SUBSTITUTION
용도: MySQL은 다양한 sql mode 방식으로 실행할 수 있으며, sql 모드에 mysql이 지원해야 하는 sql 구문, 데이터 인증 등이 정의되어 있습니다. 해당 매개변수 5.6 버전의 기본 매개변수값은 NO_ENGINE_SUBSTITUTION으로, 사용

하는 스토리지 엔진이 비활성화되었거나 Uncompiled되어 오류가 발생함을 의미합니다. 5.7 버전의 기본 매개변수값은 `ONLY_FULL_GROUP_BY` , `STRICT_TRANS_TABLES` , `NO_ZERO_IN_DATE` , `NO_ZERO_DATE` , `ERROR_FOR_DIVISION_BY_ZERO` , `NO_AUTO_CREATE_USER` , `NO_ENGINE_SUBSTITUTION` 입니다.
그중에서,
`ONLY_FULL_GROUP_BY` 는 GROUP BY 작업 시 SELECT 칼럼, HAVING 혹은 ORDER BY 칼럼이 있다면, 반드시 GROUP BY에 나타나야 하거나 GROUP BY의 함수 칼럼에 종속되어야 함을 의미합니다.
`STRICT_TRANS_TABLES` 가 포함되어 있으면 Strict mode가 활성화됩니다.
`NO_ZERO_IN_DATE` 가 날짜의 월, 일에 0을 포함하도록 허용하는지, Strict mode에 영향을 받는지 결정합니다.
`NO_ZERO_DATE` 를 포함하면 데이터베이스에서 날짜에 0을 삽입할 수 없으며, Strict mode에 영향을 받는지 결정합니다.
`ERROR_FOR_DIVISION_BY_ZERO` 는 Strict mode에서 INSERT 혹은 UPDATE 할 때 0으로 나뉘면 오류가 발생하지만 이를 알리지 않으며, Strict mode가 아닐 때 0으로 나뉘면 MySQL이 NULL을 출력합니다.
`NO_AUTO_CREATE_USER` 은 GRANT로 생성하여 비밀번호가 없는 사용자를 차단합니다.
`NO_ENGINE_SUBSTITUTION` 은 사용하는 스토리지 엔진이 비활성화되었거나 Uncompiled 되어 오류가 발생함을 의미합니다.
영향: 마스터 인스턴스에서 매개변수를 수정하면 재해 복구 인스턴스의 매개변수를 동기화 수정할 수 없습니다. 마스터 인스턴스에서 sql mode 방식은 수정하나, 재해 복구 인스턴스는 동기화 변경하지 않습니다. 예를 들어 마스터 인스턴스 sql mode 방식 제한이 재해 복구 sql mode 방식 제한보다 작을 경우, 마스터 인스턴스에서 실행 성공한 SQL을 재해 복구에 동기화할 때 오류가 발생하여 마스터/슬레이브 데이터 불일치가 발생할 수 있습니다.

MyISAM에서 InnoDB로의 자동 변환 제한

최종 업데이트 날짜: : 2024-07-25 16:38:48

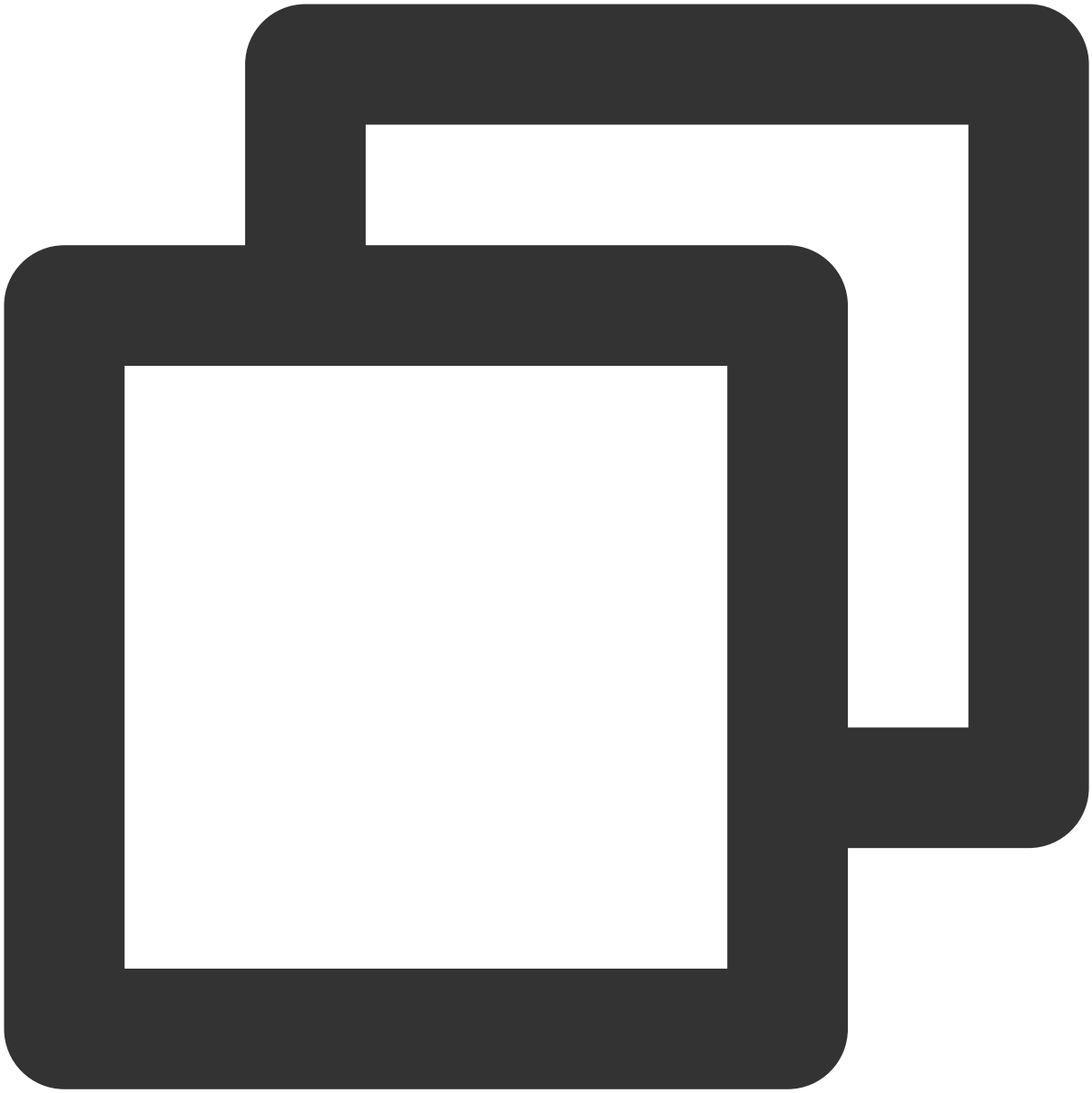
본 문서에서는 MyISAM 엔진을 InnoDB 엔진으로 자동 전환한 후 테이블 생성 시 발생하는 오류 솔루션을 소개합니다.

상황

TencentDB for MySQL에서는 기본적으로 InnoDB 스토리지 엔진을 지원하며, MySQL 5.6 및 그 이상의 버전에서는 MyISAM 엔진과 Memory 엔진을 지원하지 않습니다. 자세한 내용은 데이터베이스 스토리지 엔진을 참조하십시오. 데이터베이스를 TencentDB for MySQL 5.6 및 그 이상의 버전으로 마이그레이션 혹은 업그레이드할 경우, 시스템이 자동으로 MyISAM 엔진을 InnoDB 엔진으로 변환합니다.
MyISAM 엔진은 Auto Increment 포함 Compound Primary Key를 지원하지만 InnoDB 엔진은 지원하지 않습니다. 따라서 MyISAM 엔진을 InnoDB 엔진으로 전환하면 테이블 생성 시 오류가 발생할 수 있습니다. 오류 정보는 다음과 같습니다. `ERROR 1075 (42000):Incorrect table definition;there can be only one auto column and it must be defined as a key`
Auto Increment를 위한 인덱스를 생성하는 방식으로 InnoDB 엔진의 Auto Increment 포함 Compound Primary Key 구문을 구현하시길 권장합니다.

InnoDB 엔진에서 Auto Increment 포함 Compound Primary Key의 수정 방안

1. 기존 테이블 생성 오류 SQL 명령:

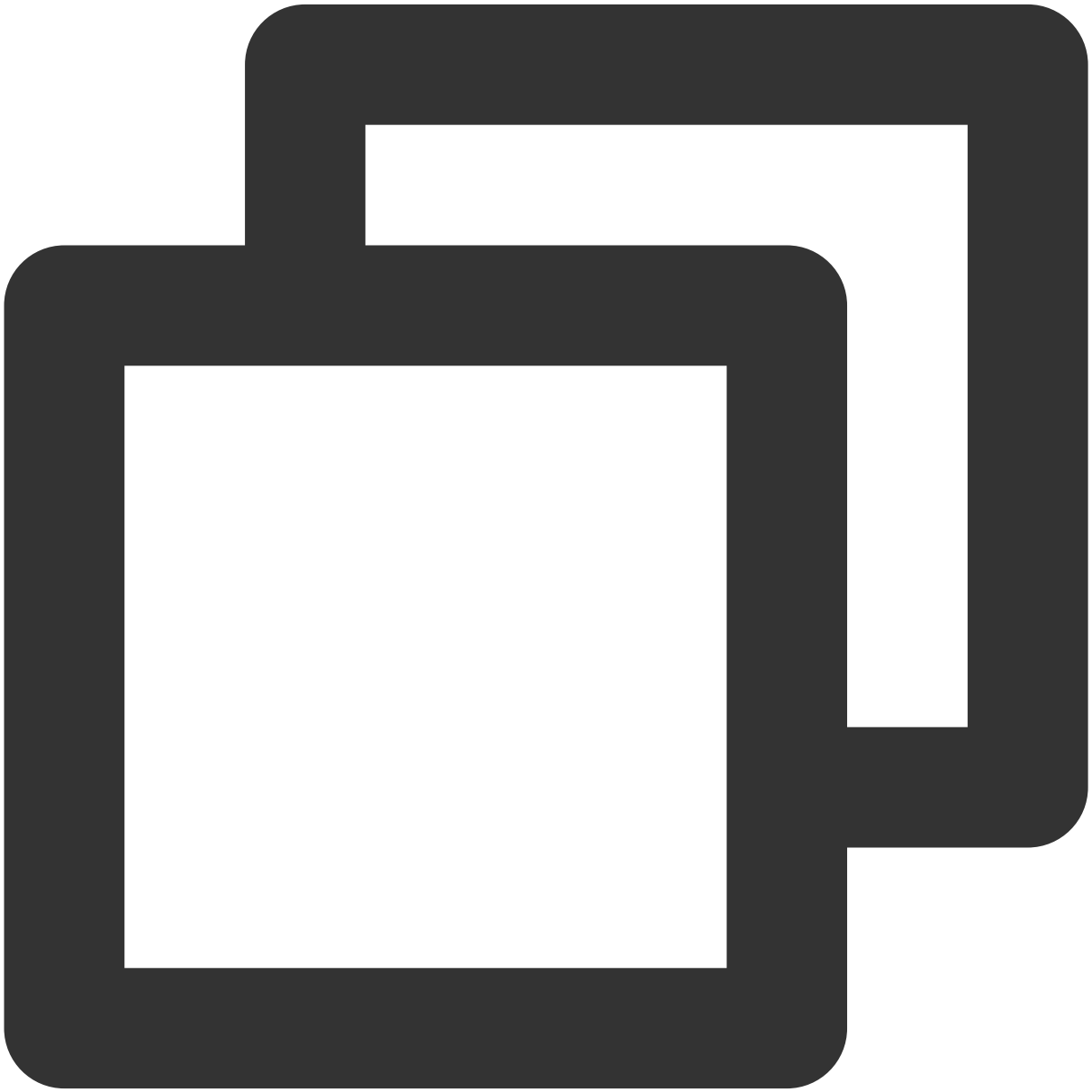


```
create table t_complexkey
(
id int(8) AUTO_INCREMENT,
name varchar(19),
value varchar(10),
primary key (name,id),
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

다음 이미지와 같이 오류 생성:

```
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 and it mus
```

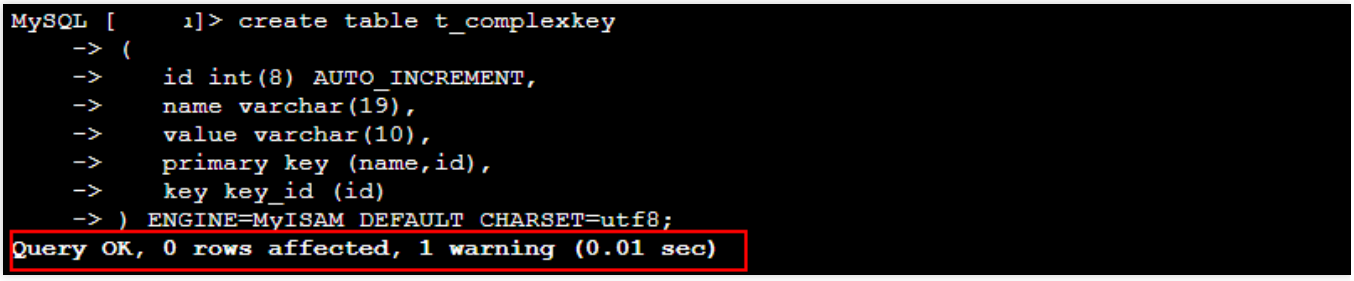
2. 인덱스 생성 수정 후의 SQL 명령:



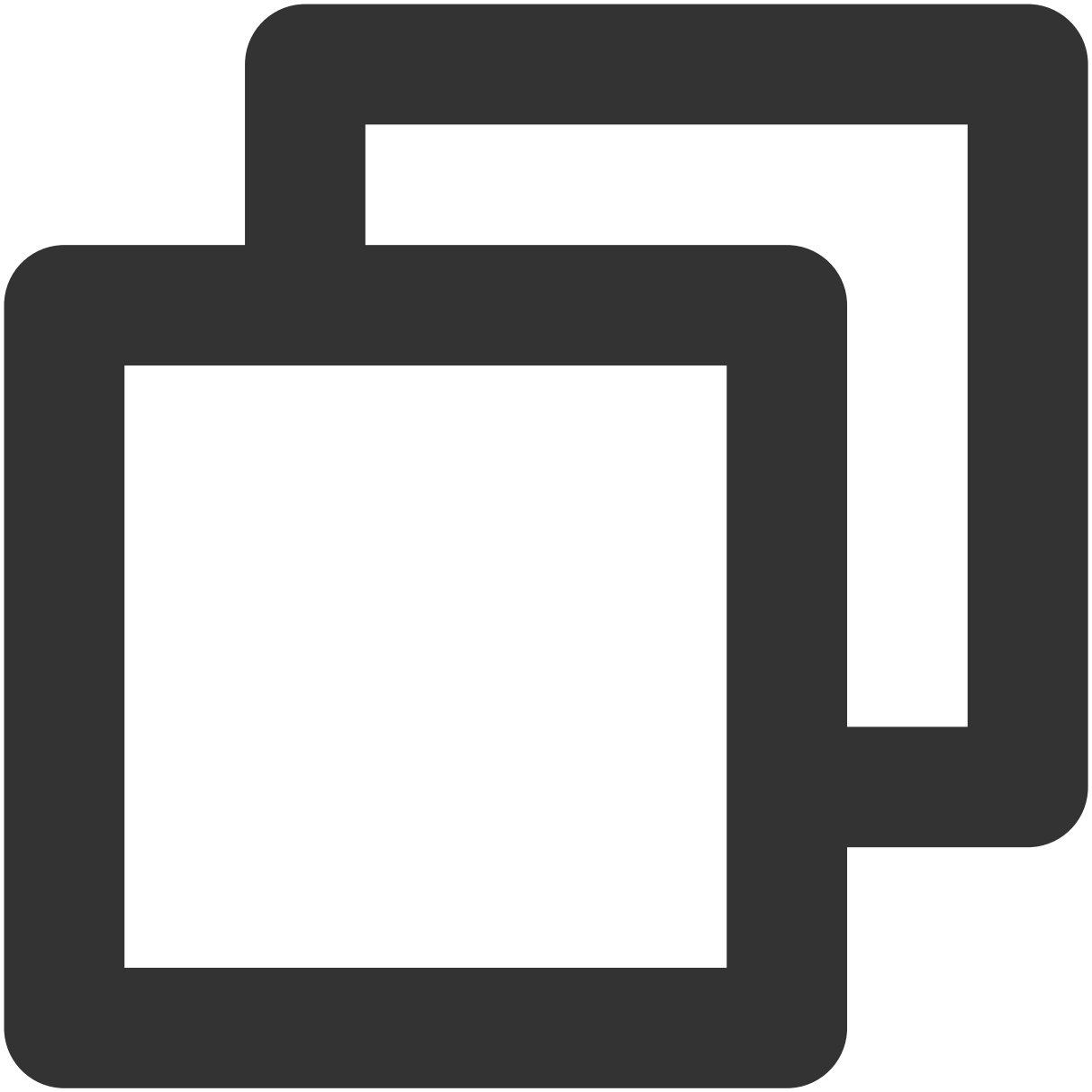
```
create table t_complexkey
(
```

```
  id int(8) AUTO_INCREMENT,
  name varchar(19),
  value varchar(10),
  primary key (name,id),
  key key_id (id)  ## 자동 증가열에 대해 인덱스 생성
  ) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

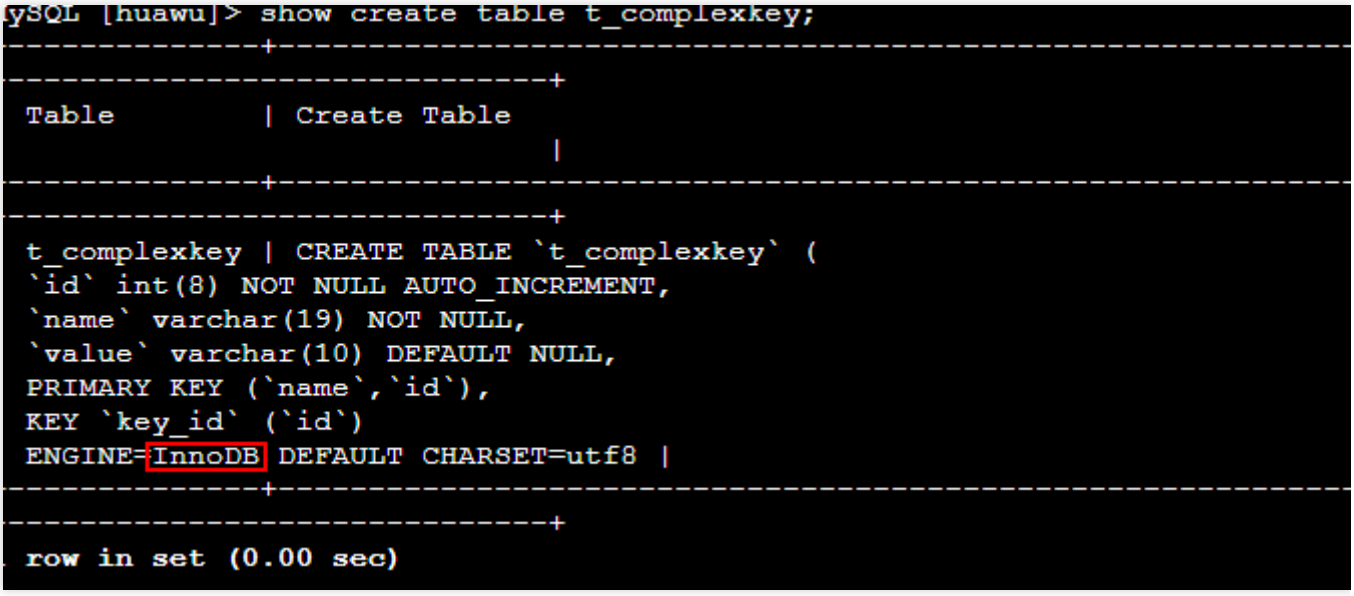
다음 이미지와 같이 생성 성공:



3. 조회 생성 후의 테이블 구조:



```
show create table t_complexkey;
```



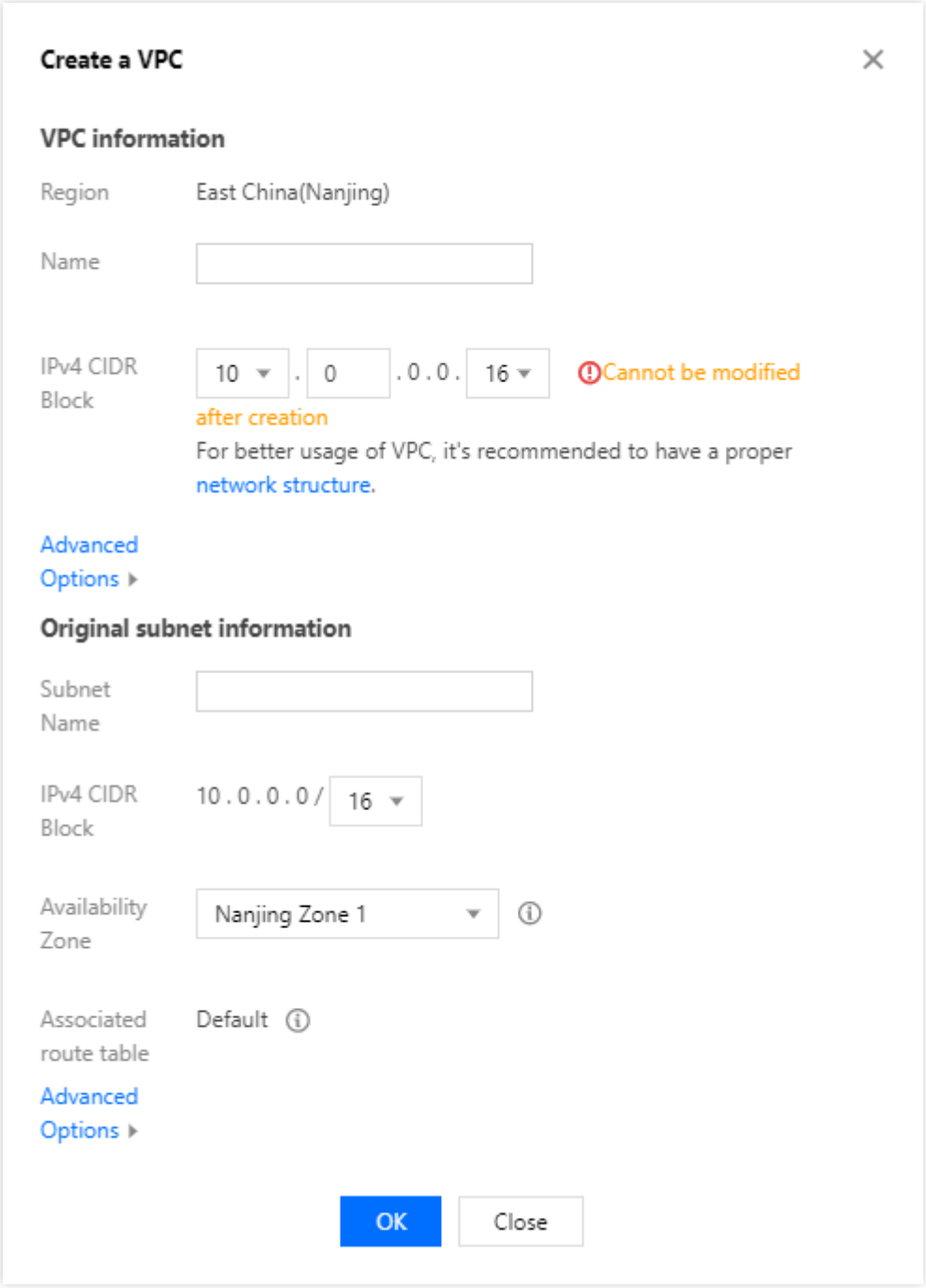
TencentDB for MySQL을 위한 VPC 생성

최종 업데이트 날짜: : 2024-07-25 16:38:48

Tencent Cloud는 클라우드 데이터베이스(CDB)의 호스팅 플랫폼: Tencent Cloud 사설 네트워크 VPC를 제공합니다. VPC에서 Tencent Cloud 리소스를 실행할 수 있습니다(예: Tencent Cloud CDB 인스턴스).
일반적인 솔루션은 동일 VPC에 있는 CDB 인스턴스와 웹서비스에서 데이터를 공유합니다. 이 튜토리얼에서는 해당 솔루션에 대해 VPC를 구축하고, CDB를 VPC에 추가하여 적용합니다.
본 문서는 동일한 VPC에서 CVM과 TencentDB for MySQL을 추가하고, VPC 내 클라우드 리소스의 내부 네트워크 통신을 구현하는 방법을 소개합니다.

1단계: VPC 생성

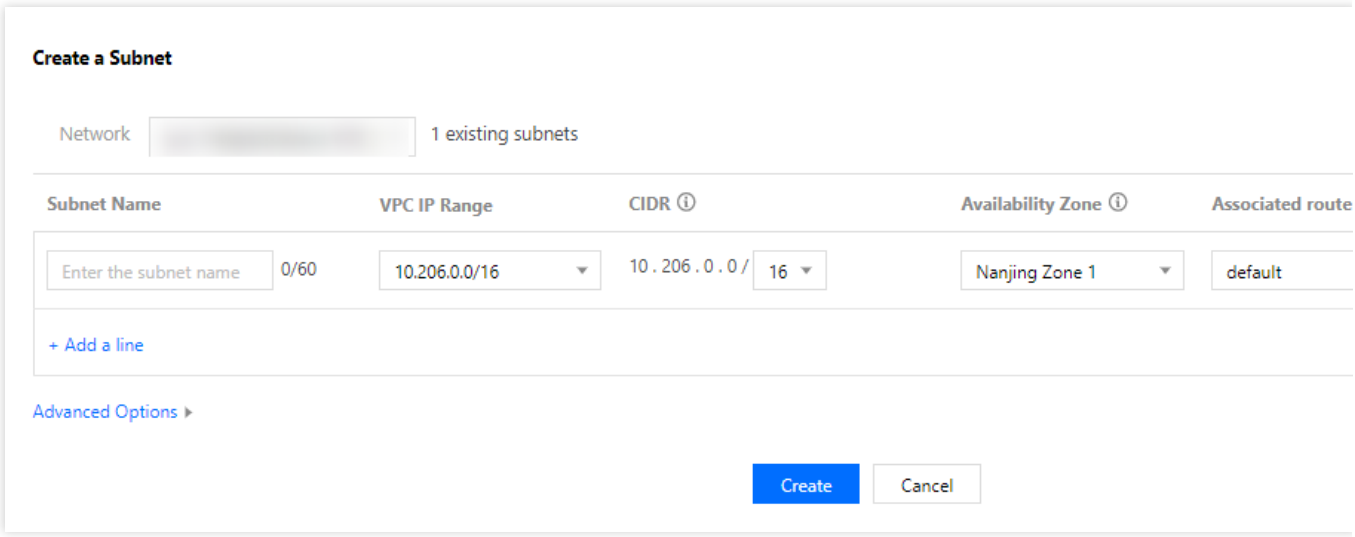
VPC는 최소 서브넷 1개를 포함해야 하고, 서브넷에서만 클라우드 서비스 리소스를 추가할 수 있습니다.
1. VPC 콘솔에 로그인합니다.
2. 리스트 위쪽에서 VPC가 속한 리전을 선택하고 **+New**를 클릭합니다.
3. VPC 정보와 초기 서브넷 정보를 작성하고 **확인**을 클릭합니다. VPC와 서브넷의 CIDR 생성 후에는 수정할 수 없습니다.
VPC CIDR은 다음 IP 대역 중 임의의 1개를 지원합니다. VPC 간 내부 네트워크 통신이 필요한 경우 PC/모바일 CIDR의 설정이 중복되지 않도록 하십시오.
`10.0.0.0` `~~10.255.255.255~~` ~~(마스크 범위 16~~28)
`172.16.0.0` `~~172.31.255.255~~` ~~(마스크 범위 16~~28)
`192.168.0.0` `~~192.168.255.255~~` ~~(마스크 범위 16~~28)
서브넷의 CIDR은 VPC의 CIDR 내에 있거나 동일해야 합니다.
예를 들어 VPC의 IP 대역이 `192.168.0.0/16` 이면, 해당 VPC 내 서브넷의 IP 대역은 `192.168.0.0/16` 、 `192.168.0.0/17` 등입니다.



2단계: 서브넷 생성

1개 혹은 멀티 서브넷을 동시에 생성할 수 있습니다.

1. VPC 콘솔에 로그인합니다.
2. 왼쪽 목록에서 **서브넷**을 클릭해 관리 페이지로 들어갑니다.
3. 생성할 서브넷의 리전과 VPC를 선택하고 **+New**를 클릭합니다.
4. 서브넷 이름, CIDR, 가용존, 연결 라우팅 테이블을 작성합니다.

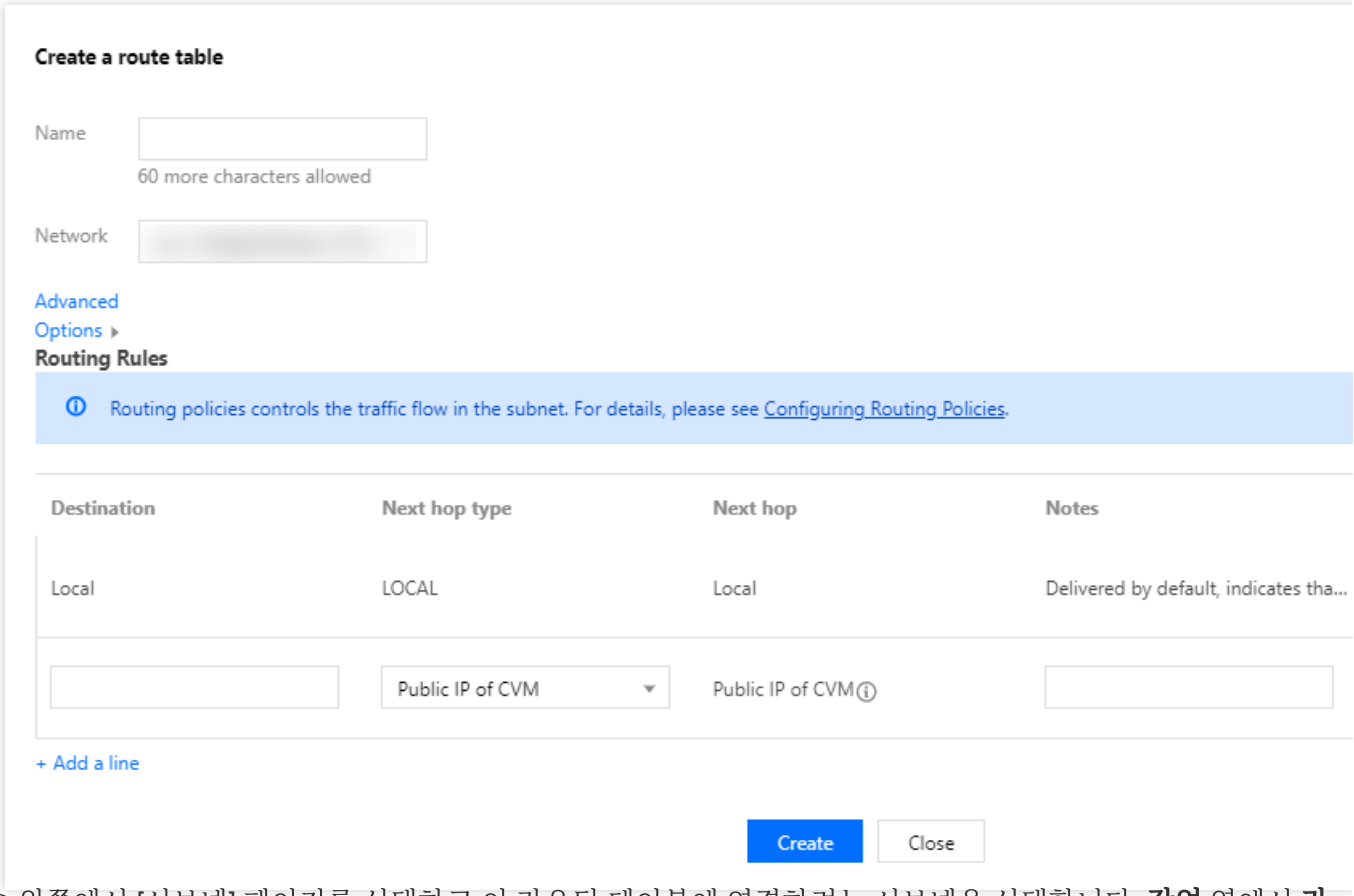


5. (옵션) [+행 추가]를 클릭하면 동시에 멀티 서브넷을 생성할 수 있습니다.
6. **생성**을 클릭합니다.

3단계: 라우팅 테이블 연결 서브넷 생성

사용자 정의 라우팅 테이블을 생성하고, 라우팅 정책을 편집하며, 지정 서브넷을 연결할 수 있습니다. 서브넷 연결 라우팅 테이블은 이 서브넷의 아웃바운드 라우팅을 지정합니다.

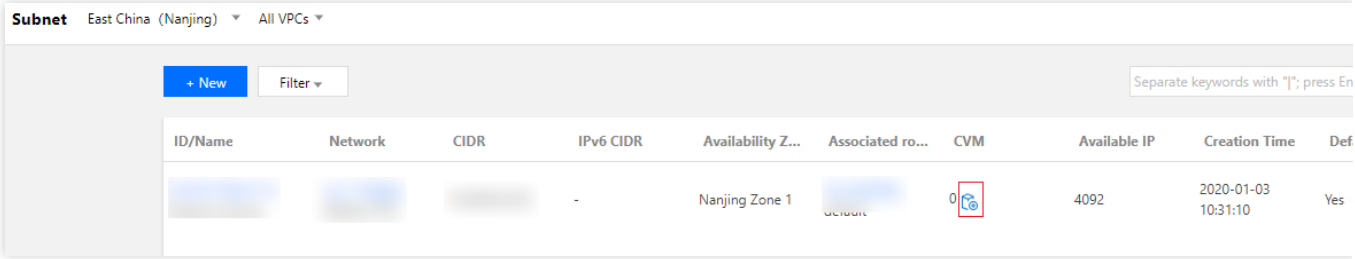
1. VPC 콘솔에 로그인합니다. 왼쪽 사이드바에서 **라우팅 테이블** 태그를 선택합니다.
2. 리스트 위쪽에서 리전과 VPC를 선택하고 **+New**를 클릭합니다.
3. 팝업 대화 상자에 이름, 소속 네트워크 및 생성 라우팅 정책을 입력하고 **생성**을 클릭합니다. 라우팅 테이블 리스트로 돌아가면 생성한 라우팅 테이블을 확인할 수 있습니다.



4. 콘솔 왼쪽에서 [서브넷] 페이지를 선택하고 이 라우팅 테이블에 연결하려는 서브넷을 선택합니다. **작업** 열에서 **라우팅 테이블 변경**을 클릭하여 연결합니다.

4단계: CVM 추가

1. VPC 콘솔에 로그인합니다.
2. 왼쪽 목록에서 **서브넷**을 클릭해 관리 페이지로 들어갑니다.
3. 추가할 CVM의 서브넷이 위치한 행에서 CVM 추가 아이콘을 클릭합니다.

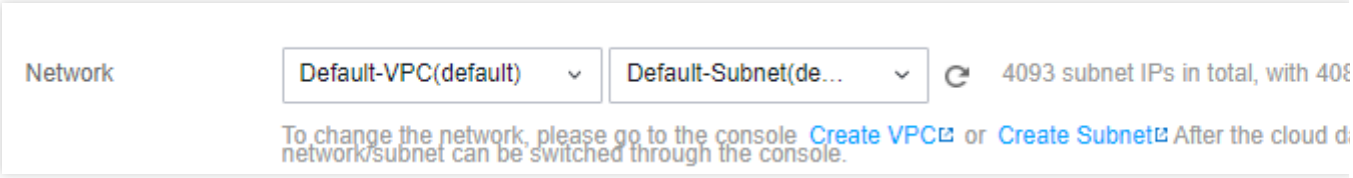


4. 페이지 안내에 따라 CVM을 구매하면 됩니다. 자세한 내용은 CVM 문서 구매 방식을 참조 바랍니다.

5단계: CDB 추가

데이터베이스 생성

1. TencentDB for MySQL 콘솔에 로그인합니다. **New**를 클릭해 구매 페이지에 접속합니다.
2. 구매 페이지의 **네트워크** 옵션에서 **사설 네트워크**를 선택합니다. 이전에 생성한 사설 네트워크 및 해당 서브넷을 선택하고, 구매한 CDB를 사설 네트워크에 추가합니다.



보유한 데이터베이스

1. 인스턴스 리스트에서 인스턴스 이름 또는 Operation 열의 **Manage**를 클릭하여 인스턴스 상세 페이지로 이동합니다.
2. 상세 페이지의 **소속 네트워크**에서 해당 VPC를 전환할 수 있습니다.

TencentDB for MySQL를 통해 비즈니스 부하 능력 향상

최종 업데이트 날짜: : 2024-07-25 16:38:48

우수한 성능과 확장 능력을 갖춘 데이터베이스로 기존 시스템의 부하 성능을 빠르게 향상할 수 있습니다. 똑같은 규모의 데이터베이스에서도 TencentDB for MySQL의 합리적인 사용으로 더 높은 QPS를 위해 데이터베이스 동시성을 크게 향상시킬 수 있습니다.

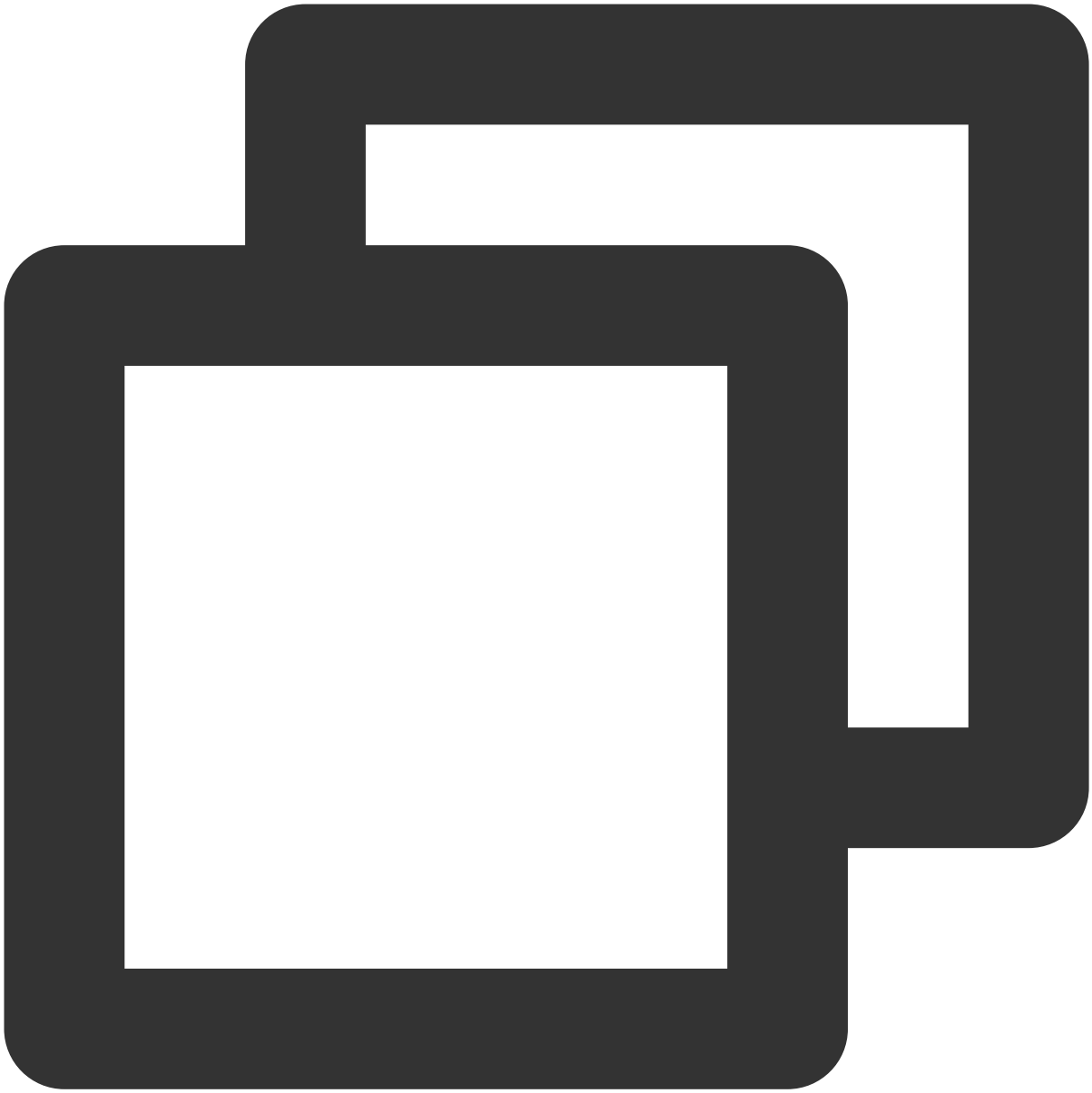
1. 적절한 데이터베이스 구성 선택

1.1 데이터베이스 버전 선택

TencentDB for MySQL은 현재 현재 v5.5, v5.6, v5.7 및 v8.0에서 사용할 수 있으며 모두 네이티브 MySQL과 완벽하게 호환됩니다. 보다 안정적인 데이터베이스 커널을 사용하고 v5.5 및 이전 버전의 디자인을 최적화하여, 더 나은 시스템 성능과 다양한 매력적인 새 기능이 제공되므로 v5.6 이상을 선택하는 것이 좋습니다.
본 문서는 MySQL v5.7을 예로 들어 새 버전의 기능을 설명합니다. 이 버전은 뛰어난 성능, 안정성 및 사용 편의성으로 널리 알려져 있습니다. 일부 개선 사항 및 새로운 기능은 다음과 같습니다.

네이티브 JSON 지원

MySQL v5.7에서는 MySQL 테이블에 네이티브 JSON 형식으로 데이터를 저장하기 위해 새로운 데이터 유형이 추가되었으며 다음과 같은 이점이 있습니다.
문서 인증: JSON 규칙에 따른 데이터 세그먼트만 JSON 유형 열에 입력할 수 있습니다. 즉, 자동화된 JSON 구문 인증이 있음을 의미합니다.
고효율 액세스: JSON 문서가 JSON 유형 열에 저장되면 데이터가 일반 텍스트로 저장되지 않습니다. 대신 최적화된 이진법 형식으로 저장되므로 객체 구성원과 배열 요소에 더 빠르게 액세스할 수 있습니다.
성능 향상: query 성능을 향상시키기 위해 JSON 형식의 열에 있는 데이터에 인덱스를 생성할 수 있습니다. 이러한 인덱스는 가상 컬럼에 생성된 '함수 인덱스'를 통해 구현할 수 있습니다.
편의성: JSON 유형 열에 연결된 인라인 구문은 features와 같은 SQL 문에서 문서 쿼리에 자연스럽게 통합될 수 있습니다. feature은 JSON 필드입니다.



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

MySQL 5.7을 사용하면 하나의 툴에서 최고의 관계형 샘플을 최고의 문서 샘플과 원활하게 통합하여 다양한 애플리케이션 및 사용 사례에서 가장 적합한 샘플을 사용하여 애플리케이션 범위를 크게 확장할 수 있습니다.

SYS Schema

MySQL SYS Schema는 뷰, 저장 프로시저, 저장 방법, 테이블 및 트리거와 같은 객체 집합으로 구성된 database schema입니다. Performance Schema 및 INFORMATION_SCHEMA의 다양한 테이블에 저장된 풍부한 모니터링 데이터에 대한 쉽고 읽기 쉬운 DBA 및 개발자 친화적인 액세스를 제공합니다.

MySQL SYS Schema는 MySQL 5.7에 기본 탑재되어 있으며, 아래와 같이 FAQ에 대한 해결을 위해 요약 뷰를 제공합니다.
데이터베이스 서비스의 모든 리소스를 차지하는 것은 무엇입니까?
데이터베이스 서버에 대한 액세스양이 가장 큰 CVM 인스턴스는 무엇인가요?
인스턴스 메모리는 어떻게 사용되나요?

InnoDB 개선

InnoDB(Online DDL)의 온라인 작업: MySQL을 다시 시작하지 않고도 Buffer Pool size를 동적으로 조정하여 비즈니스 요구 사항의 변화에 적응할 수 있습니다. InnoDB는 이제 UNDO 로그와 온라인 테이블스페이스를 자동으로 비울 수 있으므로 큰 공유 테이블스페이스 파일(ibdata1)에 대한 가장 일반적인 이유 중 하나를 제거합니다. 또한 MySQL v5.7은 인덱스 이름 변경 및 varchar 크기 변경을 지원합니다. 두 작업 모두 이전 버전에서 인덱스 또는 테이블을 재생성해야만 수행할 수 있었습니다.
InnoDB 네이티브 파티션: MySQL v5.7 InnoDB는 파티셔닝에 대한 네이티브 지원이 포함되어 있습니다. 부하를 줄이고 메모리 사용량을 최대 90%까지 낮출 수 있습니다.
InnoDB 캐시 프리패치: MySQL이 다시 시작되면 InnoDB는 버퍼 풀에서 가장 인기 있는 데이터의 25%를 자동으로 보관하므로 데이터 캐시를 미리 로딩하거나 미리 가져올 필요가 없으며 MySQL 재시작으로 인한 잠재적인 성능 손실을 방지할 수 있습니다.
MySQL 5.7의 개선 사항 및 새로운 기능에 대한 자세한 내용은 MySQL의 공식 문서를 참고하십시오.

1.2 데이터베이스 메모리 선택

현재 TencentDB for MySQL은 별도의 CPU 옵션을 제공하지 않습니다. 대신 CPU는 메모리 사양에 따라 비례적으로 할당됩니다. 비즈니스 특성에 따라 데이터베이스 사양을 구매할 수 있습니다. 사양 선택 시 참고할 수 있는 성능 정보를 제공하기 위해 각 유형의 인스턴스에 대해 철저한 벤치마크 테스트를 수행했습니다.
그러나 Sysbench 활성화 테스트가 모든 비즈니스 시나리오를 나타낼 수는 없다는 점에 유의해야 합니다. TencentDB for MySQL이 비즈니스 시나리오에서 어떻게 작동하는지 더 잘 이해할 수 있도록 인스턴스를 공식적으로 시작하기 전에 인스턴스에 대한 스트레스 테스트를 수행하는 것이 좋습니다. 자세한 내용은 성능 개요를 참고하십시오.
메모리는 디스크보다 훨씬 빠른 액세스 속도를 특징으로 하는 핵심 인스턴스 메트릭 중 하나입니다. 일반적으로 메모리에 캐시된 데이터가 많을수록 데이터베이스 응답 속도가 빨라집니다. 메모리가 작으면 저장된 데이터가 일정량을 초과한 후 초과된 데이터는 디스크에 저장됩니다. 그런 다음 새 요청이 데이터에 다시 액세스하면 디스크에서 메모리로 데이터를 읽어 디스크 IO를 소비하고 데이터베이스 응답 속도가 느려집니다.
읽기 동시성이 높거나 읽기 지연에 민감한 비즈니스의 경우 높은 데이터베이스 성능을 보장할 수 있도록 더 높은 메모리 사양을 선택하는 것이 좋습니다.

1.3 디스크 선택

TencentDB for MySQL 인스턴스의 디스크 공간에는 데이터, 시스템, binlog 및 임시 파일이 포함됩니다. 입력한 데이터의 양이 인스턴스의 디스크 용량을 초과하는 경우 인스턴스가 업그레이드되지 않으면 인스턴스 잠금이 트리거될 수 있습니다. 따라서 디스크를 구매할 때 향후 데이터 볼륨 증가 가능성을 고려하고 더 큰 디스크를 선택하여 디스크 용량 부족으로 인해 인스턴스가 잠기거나 자주 업그레이드되는 것을 방지할 수 있습니다.

1.4 적합한 데이터 복제 방식 선택

TencentDB for MySQL은 비동기화, 반동기화, 강제 동기화 세 가지 복제 방식을 제공하며, 자세한 내용은 데이터베이스 인스턴스 복사를 참고하십시오. 비즈니스가 쓰기 대기 시간 또는 데이터베이스 성능에 민감한 경우 비동기 복제 방식을 선택하는 것이 좋습니다.

1.5 TencentDB의 고가용성

TencentDB for MySQL의 고가용성은 원본 복제 아키텍처에 의해 보장됩니다. 원본-복제본 데이터 동기화는 binlog를 통해 이루어집니다. 또한 임의의 이전 시점으로 데이터베이스를 롤백할 수 있습니다. 이 기능은 백업 및 로그에 의존하므로 일반적으로 백업 및 복구 시스템을 직접 설정하거나 인스턴스의 고가용성을 유지하기 위해 추가 비용을 지불할 필요가 없습니다.

1.6 TencentDB의 확장성

TencentDB for MySQL의 모든 다양한 데이터베이스 버전과 메모리/디스크 사양은 핫 업그레이드를 지원합니다. 업그레이드 프로세스는 비즈니스를 방해하지 않으므로 비즈니스 성장으로 인한 데이터베이스 병목 현상에 대한 우려를 제거합니다.

1.7 CVM과 TencentDB for MySQL 함께 사용

구매 후 CVM과 TencentDB for MySQL을 함께 사용해야 하는 경우가 많습니다. 자세한 내용은 MySQL 인스턴스 연결을 참고하십시오.

2. 읽기 전용 인스턴스를 읽기 확장으로 사용

일반적인 인터넷 기반 비즈니스에서 데이터베이스의 읽기/쓰기 비율은 일반적으로 4:1에서 10:1 범위이며, 이는 데이터베이스의 읽기 부하가 쓰기 부하보다 훨씬 높다는 것을 의미합니다. 성능 병목 현상이 발생하면 일반적인 솔루션은 읽기 로드를 처리하는 기능을 향상시키는 것입니다.
TencentDB for MySQL 읽기 전용 인스턴스는 이러한 문제에 읽기 확장 솔루션을 제공합니다. 자세한 내용은 읽기 전용 인스턴스 생성을 참고하십시오.
읽기 전용 인스턴스는 다양한 비즈니스에서 읽기 전용 액세스에 사용할 수도 있습니다. 예를 들어 원본 인스턴스는 온라인 비즈니스에 대한 읽기/쓰기 액세스를 수행하는 반면 읽기 전용 인스턴스는 내부 비즈니스 또는 데이터 분석 플랫폼에 대한 읽기 전용 쿼리를 제공합니다.

3. TencentDB 재해 복구 솔루션

TencentDB for MySQL은 재해 복구 인스턴스를 제공하여 데이터베이스에 대한 원격 재해 복구를 신속하게 설정할 수 있도록 도와줍니다.

재해 복구 인스턴스의 도움으로 서로 다른 리전에 있는 여러 데이터 센터가 서로 중복으로 작동할 수 있으므로 한 데이터 센터가 장애 또는 불가항력 이벤트로 인해 서비스를 제공할 수 없는 경우 서비스를 다른 데이터 센터로 신속하게 전환할 수 있습니다. 재해 복구 인스턴스는 Tencent Cloud 사설망을 사용하여 데이터를 동기화하고 복제는 MySQL 커널 수준에서 최적화되어 재해 발생 시 지연된 동기화가 비즈니스에 미치는 영향을 최소화할 수 있습니다. 원격 서비스 논리가 준비되어 있는 한 재해 복구 전환은 몇 초 안에 완료될 수 있습니다.

4. 2리전 3데이터센터 프로그램

TencentDB for MySQL을 사용하면 몇 가지 간단한 단계만으로 2리전 3데이터센터 스키마를 구성할 수 있습니다. TencentDB for MySQL 도시 내 강력한 일관성 클러스터를 구입하고 1리전 2데이터센터 용량을 제공하는 멀티 가용존 배포(현재 베타 테스트 중)를 선택합니다.
2리전 3데이터센터 아키텍처를 구축하기 위해 클러스터에 원격 재해 복구 노드를 추가합니다.

5. 재해 복구 인스턴스를 사용하여 사용자에게 근거리 액세스 제공

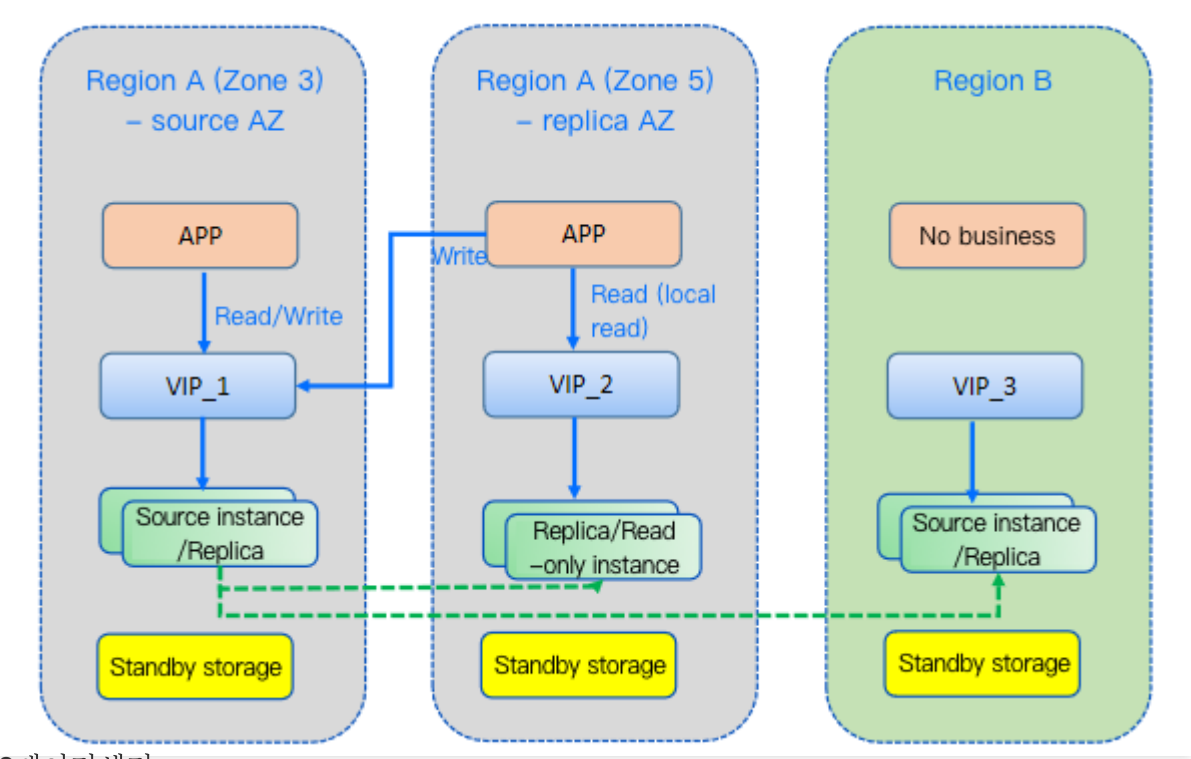
재해 복구 인스턴스는 또한 고가용성 M-S(원본 복제) 아키텍처를 채택합니다. 또한 읽기 전용 방식으로 액세스할 수 있으므로 다른 리전의 사용자가 가까운 곳의 비즈니스에 액세스해야 하는 시나리오에서도 안심하고 재해 복구 인스턴스를 사용할 수 있습니다.

2리전 3데이터센터 재해 복구 아키텍처 구축

최종 업데이트 날짜: : 2024-07-25 16:38:38

본 문서에서는 가용존(AZ) 전체에 인스턴스를 배포하고 원격 재해 복구 인스턴스를 생성하여 2리전 3데이터센터 아키텍처를 설정하는 방법을 설명합니다.

2리전 3데이터센터 배포 아키텍처



1리전 2데이터센터:

리전 A의 3존과 5존은 1리전 2데이터센터 아키텍처를 형성합니다. 3존이 실패하면 5존으로 전환하여 데이터베이스를 보호할 수 있습니다.

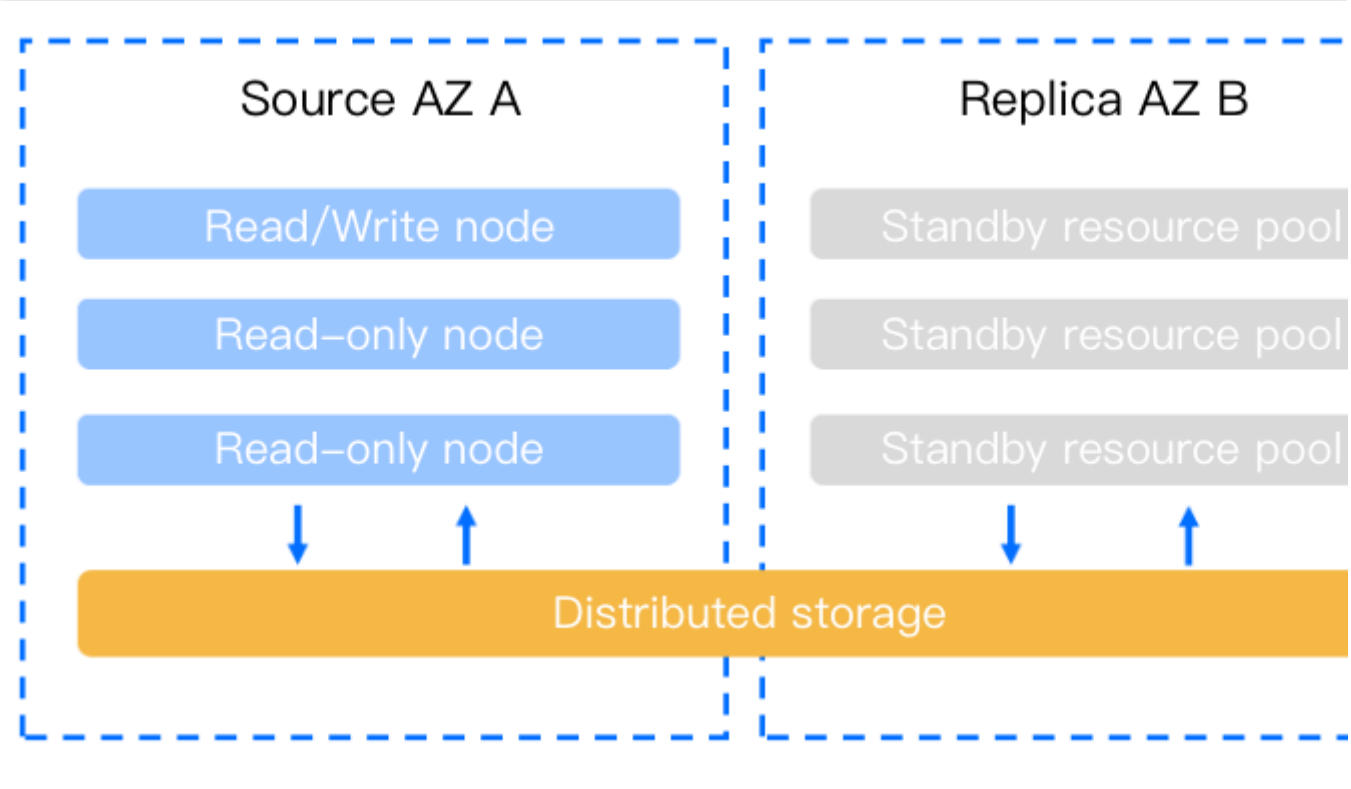
크로스 리전 재해 복구:

리전 A와 B는 교차 지역 재해 복구 아키텍처를 형성합니다. A 리전의 3, 5존에 있는 모든 데이터 센터에 장애가 발생하더라도 B 리전의 데이터 센터로 전환한 후 비즈니스를 계속할 수 있습니다.

다중 AZ 배포

TencentDB for MySQL는 다중 AZ 배포를 지원합니다. 단일 AZ 배포 체계와 비교할 때 다중 AZ 배포 체계는 재해 복구 기능이 더 우수하고 데이터베이스 인스턴스 오류, AZ 중단 및 IDC 수준 오류의 영향을 받지 않도록 데이터베이스를 보호할 수 있습니다.

TencentDB for MySQL에서는 여러 AZ가 단일 다중 AZ로 결합되어 데이터베이스 인스턴스의 고가용성 및 장애 조치 기능을 보장합니다.



전제 조건

인스턴스가 실행 중입니다.
인스턴스가 있는 리전에는 최소 2개의 AZ가 있어야 합니다.
대상 AZ에 충분한 컴퓨팅 리소스가 있습니다.

지원되는 리전 및 AZ

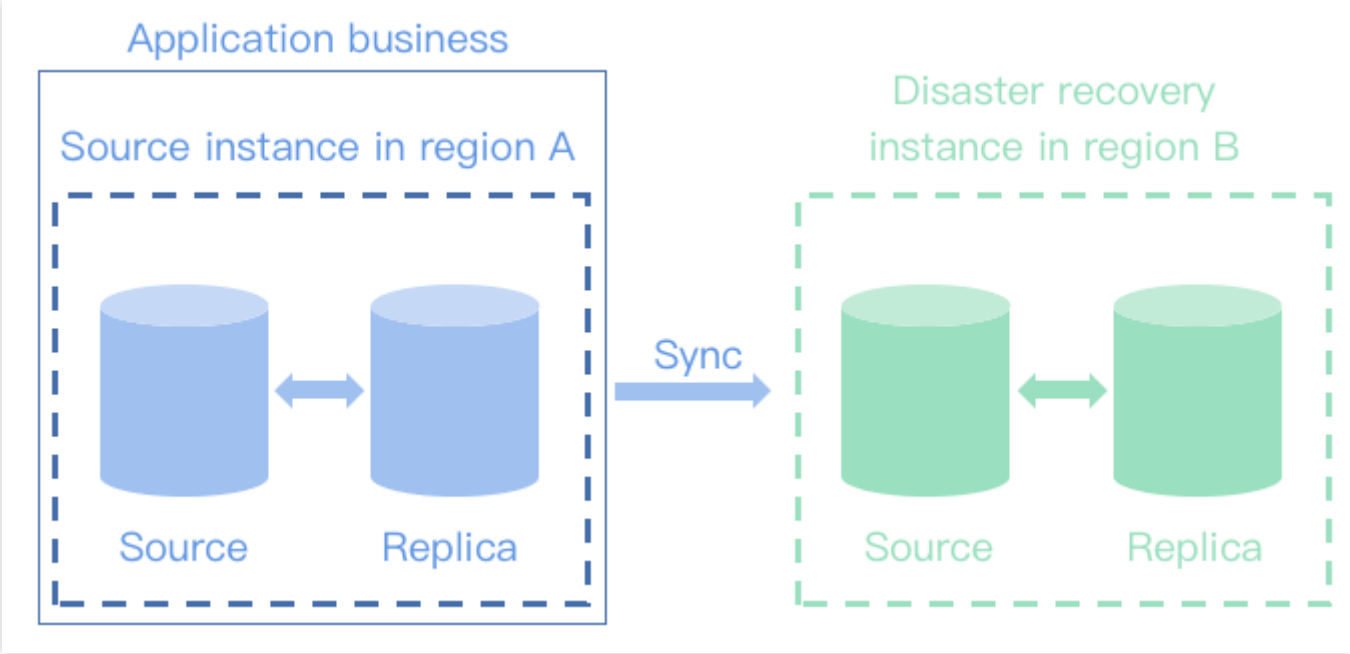
현재 TencentDB for MySQL의 다중 AZ 배포는 광저우, 상하이, 난징, 베이징, 청두, 중국홍콩, 싱가포르, 자카르타, 방콕, 뭄바이, 서울, 도쿄, 버지니아 및 프랑크푸르트 리전에서 지원됩니다. 다른 리전에서 사용 가능한 원본 및 복제본 AZ는 TencentDB for MySQL 구매 페이지에 표시된 대로입니다.
이 기능은 점차 더 많은 리전과 AZ를 지원할 예정입니다.
비즈니스 요구 사항에 맞게 다른 리전이나 AZ에 인스턴스를 배포하려면 티켓 제출을 통해 신청하십시오.

요금 설명

이 기능은 무료입니다. 단일 AZ에서 여러 AZ로 인스턴스를 마이그레이션하는 데는 요금이 부과되지 않습니다.

크로스 리전 재해 복구 인스턴스

서비스 연속성, 데이터 안정성 및 규정 준수에 대한 요구 사항이 높은 애플리케이션의 경우 TencentDB for MySQL은 리전 간 재해 복구 인스턴스를 제공하여 낮은 비용으로 지속적인 서비스를 제공하고 데이터 안정성을 개선할 수 있는 기능을 향상하는 데 도움이 됩니다.
별도의 데이터베이스 연결 주소를 사용하여 크로스 리전 재해 복구 인스턴스는 더 낮은 비용의 장치 이중화로 근거리 액세스 및 데이터 분석과 같은 다양한 시나리오에 대한 읽기 액세스 기능을 제공할 수 있습니다. 고가용성 원본/복제본 아키텍처는 데이터베이스의 단일 실패 지점을 방지하는 데 도움이 됩니다.



요금 설명

재해 복구 인스턴스의 비용은 연결된 원본 인스턴스와 동일합니다.

2리전 3데이터센터 아키텍처 설정

1단계: 다중 AZ 배포 설정

인스턴스 구매 시 다중 AZ 배포 선택

1. TencentDB for MySQL 콘솔에 로그인한 후 인스턴스 리스트에서 **생성**을 클릭하여 구매 페이지로 이동합니다.
2. 구매 페이지에서 지원되는 리전을 선택한 다음 **복제본 AZ**에 대해 원본 AZ와 다른 AZ를 선택합니다.

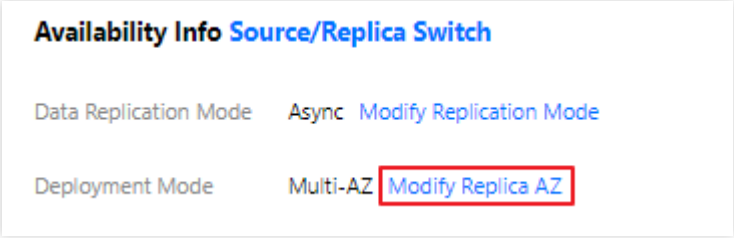
설명 :

원본/복제본이 서로 다른 AZ에 있는 경우 네트워크 동기화 지연이 2~3ms 증가할 수 있습니다.

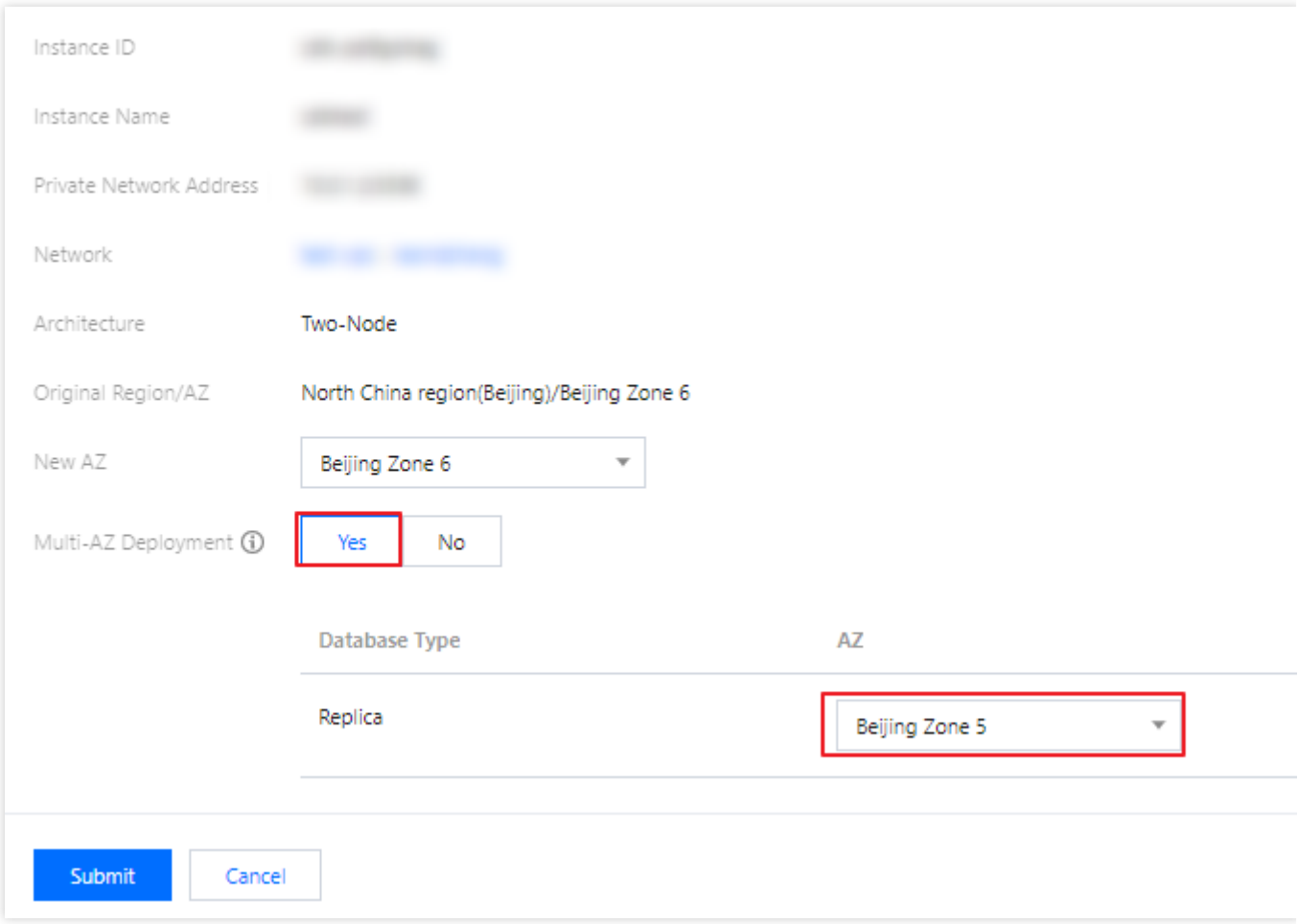
3. 모든 항목이 올바른지 확인한 후 **즉시 구매**를 클릭합니다. 결제 후 **인스턴스 세부 정보 페이지 > 가용성 정보**에서 인스턴스의 원본 및 복제본 AZ를 볼 수 있습니다.

기존 인스턴스의 단일 AZ 배포를 다중 AZ 배포로 변경

1. TencentDB for MySQL 콘솔에 로그인합니다.
2. 인스턴스 리스트에서 대상 인스턴스의 **작업** 열에 있는 ID 또는 **관리**를 클릭하여 인스턴스 세부 정보 페이지로 이동합니다.
3. 인스턴스 세부 정보 페이지의 **가용성 정보 > 배포 모드**에서 **복제본 AZ 수정**을 클릭합니다.



4. 팝업 창에서 다중 AZ 배포에 대해 **예**를 선택하고 복제본 AZ를 선택한 다음 **제출**을 클릭합니다.



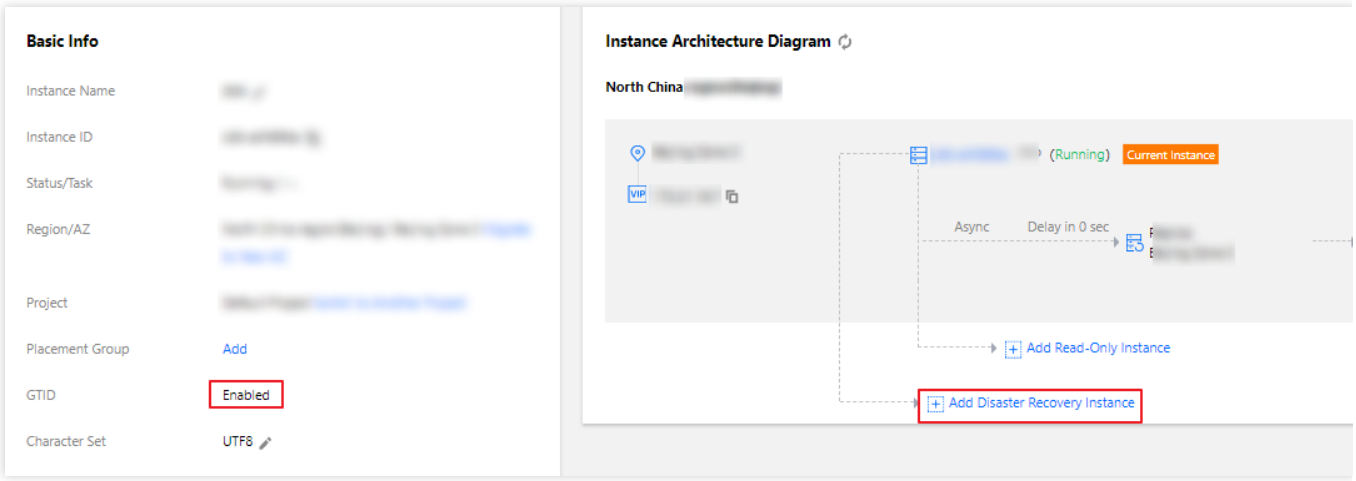
2단계: 크로스 리전 재해 복구 인스턴스 설정

설명：

재해 복구 인스턴스는 1GB 메모리 및 50GB 디스크 용량 이상 사양의 InnoDB 엔진을 사용하는 MySQL 5.6 이상의 고가용성 GTID 지원 원본 인스턴스에 대해서만 구입할 수 있습니다. 원본 인스턴스가 이 사양보다 낮으면 먼저 업그레이드하십시오.

2.1 재해 복구 인스턴스 생성

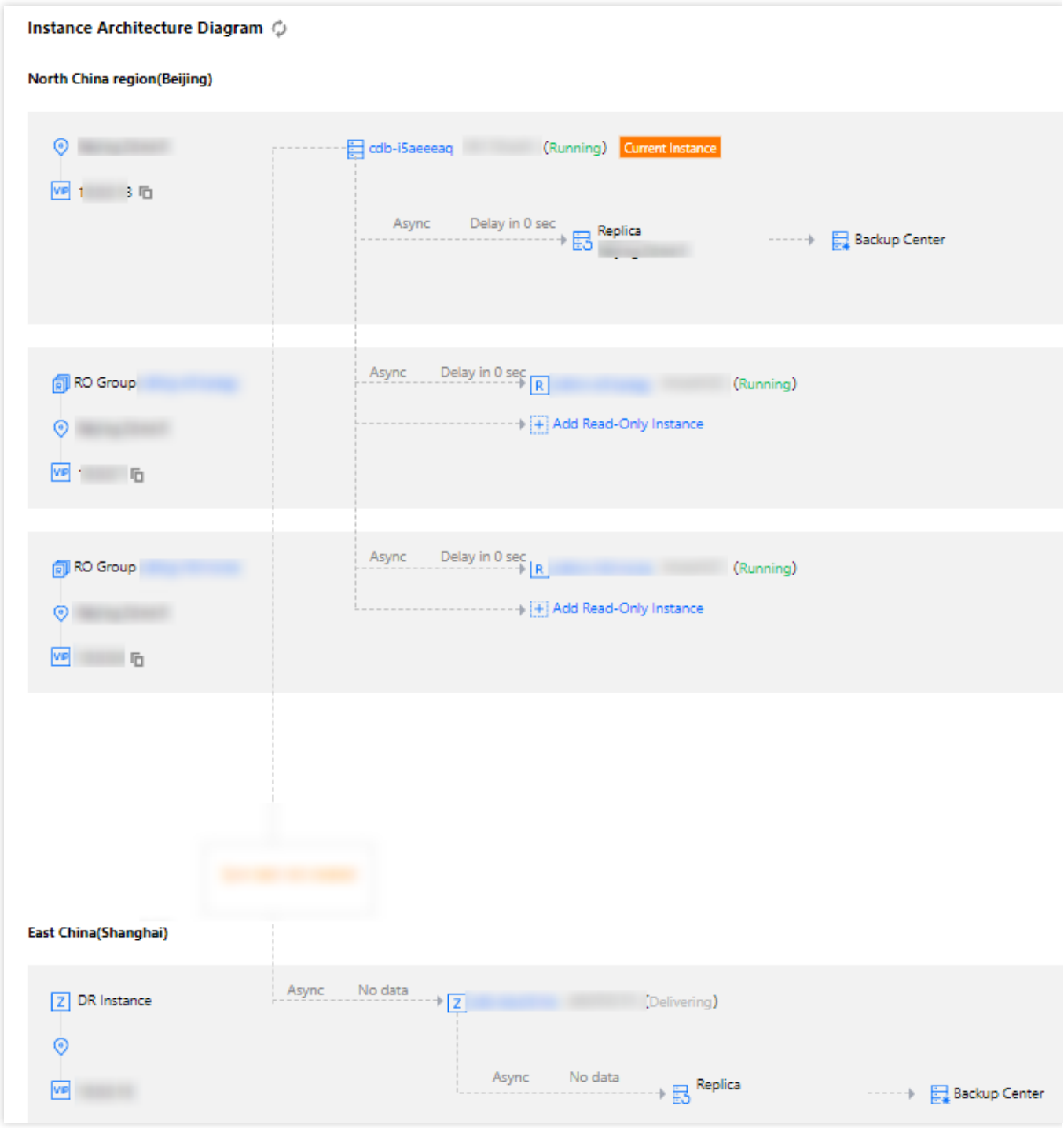
1. TencentDB for MySQL 콘솔에 로그인한 후, 인스턴스 리스트에서 인스턴스 ID 또는 **작업** 열의 **관리**를 클릭하여 세부 정보 페이지로 이동합니다.
2. 인스턴스 상세 페이지에서 인스턴스의 기본 정보를 확인하여 GTID 기능이 활성화되어 있는지 확인합니다. 인스턴스 아키텍처 다이어그램에서 **재해 복구 인스턴스 추가**를 클릭하여 재해 복구 인스턴스 구매 페이지로 이동합니다.



3. 구매 페이지에서 청구 모드, 리전 및 동기화 정책과 같은 재해 복구 인스턴스의 기본 정보를 구성합니다.
동기화 정책이 **즉시 동기화**인 경우 재해 복구 인스턴스가 생성되는 즉시 데이터가 동기화됩니다.
동기화 정책이 **생성 후 동기화**인 경우 인스턴스가 성공적으로 생성된 후 재해 복구 동기화 링크를 구성해야 합니다. 자세한 사용법은 아래의 동기화 링크 생성을 참고하십시오.
설명：
생성을 완료하는 데 필요한 시간은 데이터 양에 따라 다르며 생성 중에는 콘솔의 원본 인스턴스에 대해 어떤 작업도 수행할 수 없습니다. 적절한 시기에 하는 것이 좋습니다.
현재 전체 인스턴스 데이터만 동기화할 수 있습니다. 디스크 공간이 충분한지 확인하십시오.
원본 인스턴스가 실행 상태이고 구성 조정 작업, 다시 시작 작업 및 기타 수정 작업이 실행되지 않았는지 확인해야 합니다. 그렇지 않으면 동기화 작업이 실패할 수 있습니다.
4. 모든 항목이 올바른지 확인한 후 **즉시 구매**를 클릭하고 재해 복구 인스턴스가 전달될 때까지 기다립니다.
5. 인스턴스 리스트로 되돌아갑니다. 인스턴스 상태가 **실행 중**으로 변경되면, 정상적으로 사용할 수 있습니다.

2리전 3데이터센터 아키텍처 보기

구성이 완료되면 아래와 같이 데이터베이스가 2리전 3데이터센터 아키텍처로 배포됩니다.

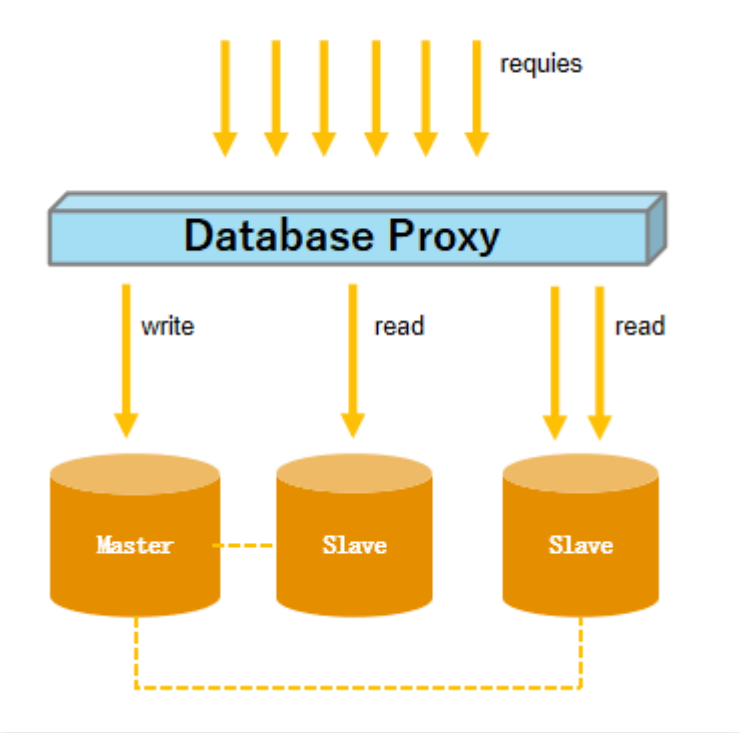

Instance Architecture Diagram
North China region(Beijing)
cdb-i5aeeeaq
(Running)
Current Instance
Async
Delay in 0 sec
Replica
Backup Center
RO Group
Async
Delay in 0 sec
(Running)
Add Read-Only Instance
RO Group
Async
Delay in 0 sec
(Running)
Add Read-Only Instance
East China(Shanghai)
DR Instance
Async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

읽기/쓰기 분리로 TencentDB for MySQL 성능 향상

최종 업데이트 날짜: : 2024-07-25 16:38:48

본 문서에서는 데이터베이스 프록시 서비스를 통해 읽기/쓰기 분리 기능을 활성화하여 수평 확장을 구현하고 TencentDB for MySQL의 성능을 향상시키는 방법을 설명합니다.

데이터베이스 프록시를 통한 읽기/쓰기 분리 아키텍처 구현



데이터베이스 프록시

데이터베이스 프록시는 TencentDB 서비스와 애플리케이션 서비스 사이에 위치한 네트워크 프록시 서비스입니다. 애플리케이션 서비스가 TencentDB에 액세스할 때 발생하는 모든 요청을 중계하는 데 사용됩니다.
데이터베이스 프록시 액세스 주소는 원래 데이터베이스 액세스 주소와 무관합니다. 프록시 주소에 도달하는 요청은 모두 프록시 클러스터를 통해 릴레이되어 데이터베이스의 원본 및 복제본 노드에 액세스합니다. 읽기/쓰기 요청이 분리되어 읽기 요청이 읽기 전용 인스턴스로 전달되어 원본 데이터베이스의 부하가 줄어듭니다.

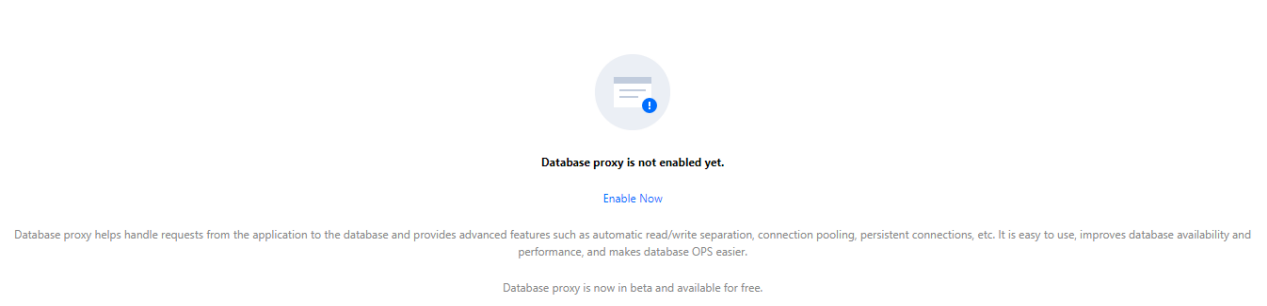
자동 읽기/쓰기 분리

귀하의 비즈니스는 예측 불가능한 비즈니스 부하뿐만 아니라 더 많은 읽기, 더 적은 쓰기 시나리오에 직면할 수 있습니다. 읽기 요청이 많은 애플리케이션 시나리오에서는 단일 인스턴스가 부하를 견디지 못하여 잠재적으로 비즈니스에 영향을 줄 수 있습니다.
읽기 기능의 자동 크기 조정을 구현하고 데이터베이스 부하를 완화하기 위해 하나 이상의 읽기 전용 인스턴스를 만들고 이를 사용하여 많은 수의 데이터베이스 읽기를 유지할 수 있습니다. 그러나 이 솔루션은 비즈니스가 읽기/쓰기 분리를 지원하도록 변환될 수 있어야 하며 코드 견고성은 비즈니스 읽기/쓰기 분리의 품질을 결정하므로 높은 기술 요구 사항을 필요로 하며 유연성과 확장성이 낮습니다.
읽기 전용 인스턴스를 생성한 후 데이터베이스 프록시 서비스를 구매하여 읽기/쓰기 분리 기능을 활성화할 수 있습니다. 그런 다음, 쓰기 요청을 원본 인스턴스에 자동으로 전달하고 읽기 요청을 읽기 전용 인스턴스에 전달하도록 애플리케이션에서 데이터베이스 프록시 주소를 구성할 수 있습니다.

데이터베이스 프록시를 통한 읽기/쓰기 분리 활성화

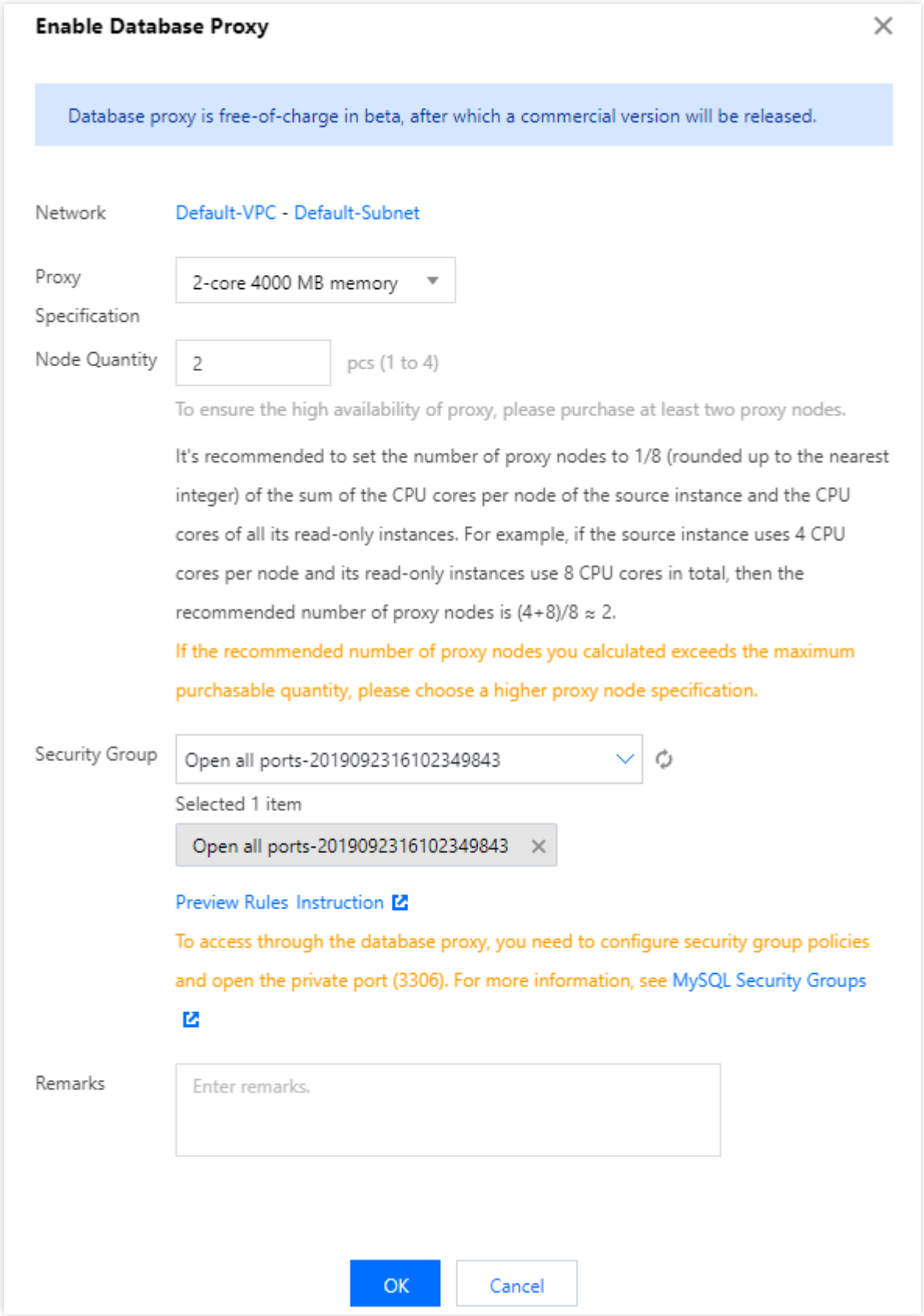
1단계: 데이터베이스 프록시 활성화

1. TencentDB for MySQL 콘솔에 로그인한 뒤, 인스턴스 리스트에서 프록시를 활성화할 원본 인스턴스를 선택하고, 인스턴스 ID 혹은 **작업**열의 **관리**를 클릭하여 인스턴스 관리 페이지로 이동합니다.
2. 인스턴스 관리 페이지에서 **데이터베이스 프록시** 탭을 선택하고 **지금 활성화**를 클릭합니다.



3. 팝업 창에서 사양 노드를 선택하고 **확인**을 클릭한 다음 페이지를 새로고침 합니다.
네트워크 유형: VPC만 지원하며, 기본 설정은 원본 인스턴스와 동일합니다.
프록시 사양: 2코어 4000MB 메모리, 4코어 8000MB 메모리, 8코어 16000MB 메모리.
노드 개수: 프록시 노드 수. 프록시 노드 수는 원본 인스턴스와 읽기 전용 인스턴스 CPU 코어 수 합의 1/8(올림한 정수)로 설정을 권장합니다. 예: 원본 인스턴스의 CPU가 4코어이고, 읽기 전용 인스턴스의 CPU는 8코어일 경우, 권장 프록시 수 = (4 + 8) / 8 ≈ 2
연결 풀 상태: 연결 풀은 비영구적 연결 비즈니스에서 빈번한 새 연결로 인해 발생하는 과도하게 높은 데이터베이스 인스턴스 부하를 완화할 수 있습니다.

보안 그룹: 네트워크 보안 격리의 중요한 수단이며 필요에 따라 기존 보안 그룹을 선택하거나 새 보안 그룹을 생성할 수 있습니다.

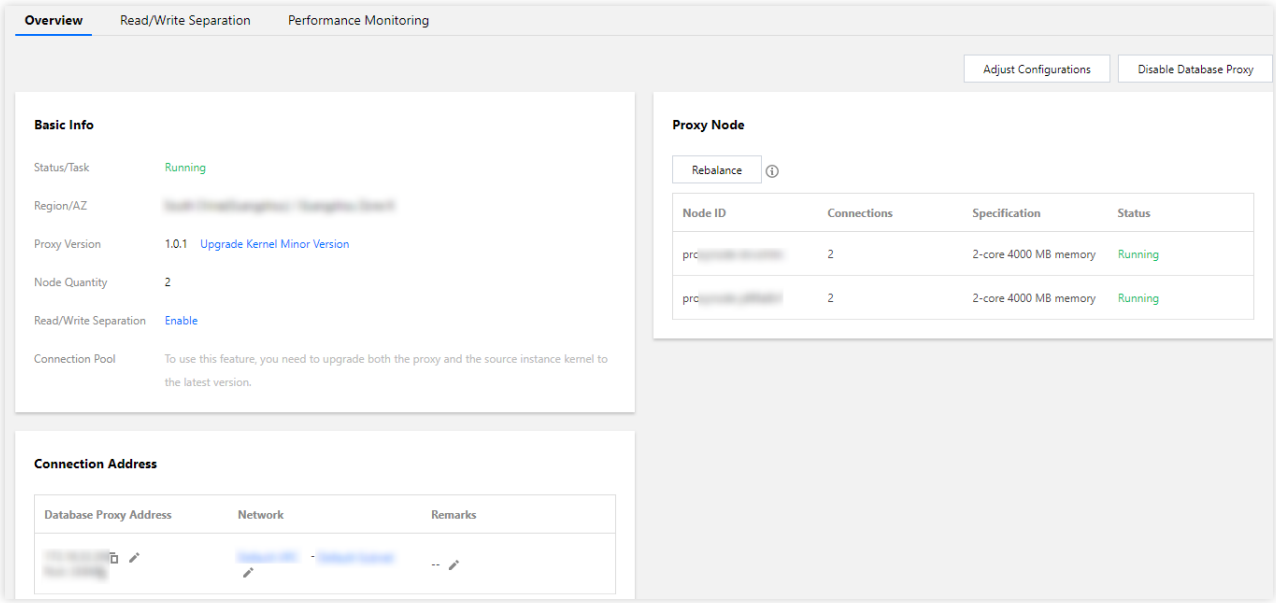


4. 서비스를 성공적으로 활성화한 후 데이터베이스 프록시 페이지에서 기본 정보 조회, 프록시 노드를 관리, 데이터베이스 프록시 주소 수정, 구성 조정 등을 수행할 수 있습니다.

설명：

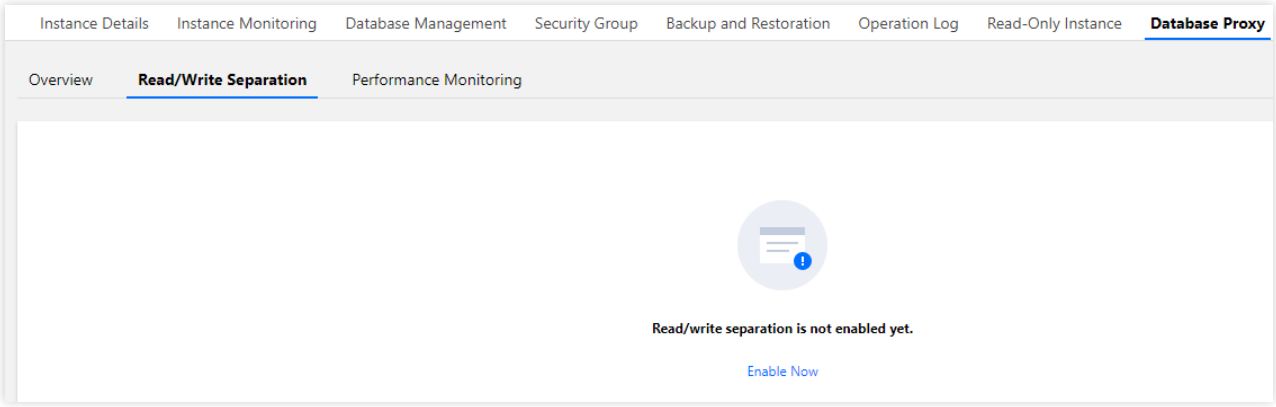
프록시 노드 목록에서 **연결 수**를 확인하거나 각 프록시 노드의 성능 모니터링 데이터를 확인하여 노드의 연결 수가 불균형한지 확인할 수 있으며, 불균형한 경우 **리밸런싱**을 클릭하여 연결을 분산할 수 있습니다.

리밸런싱으로 인해 프록시 노드가 다시 시작되고 다시 시작하는 동안 서비스를 일시적으로 사용할 수 없게 됩니다. 사용량이 적은 시간에 서비스를 다시 시작하는 것이 좋습니다. 귀하의 비즈니스에 재연결 메커니즘이 있는지 확인하십시오.



2단계: 데이터베이스 프록시 읽기/쓰기 분리 활성화

1. MySQL 콘솔에 로그인합니다.
2. 위에서 리전을 선택하고 인스턴스 ID 및 **작업** 열에서 **관리**를 클릭하여 인스턴스 관리 페이지로 이동합니다.
3. 인스턴스 관리 페이지의 **데이터베이스 프록시** 페이지에서 **읽기/쓰기 분리** 페이지를 선택하고 **지금 활성화**를 클릭합니다.



4. 팝업 창에서 관련 설정을 선택하고 **확인**을 클릭합니다.

주의 :

실행 상태에 있는 원본 및 읽기 전용 인스턴스만 데이터베이스 프록시에 추가할 수 있습니다.

현재 원격 RO와 딜레이 RO는 데이터베이스 프록시에 마운트 할 수 없습니다.

읽기 전용 인스턴스 딜레이 제거: 제거 정책 실행 여부. 읽기 전용 인스턴스에서 복사 이상 경고가 발생(복사 딜레이, 복사 중단)할 경우, 데이터베이스 프록시는 읽기 전용 인스턴스의 읽기/쓰기 분리를 일시적으로 삭제합니다. 딜레이 제거 임계값의 기본 설정은 10초이며, 읽기 전용 인스턴스의 최소 보관 수량은 1개 입니다.

설명 :

읽기 전용 인스턴스 제거 임계값과 최소 보관 수량을 설정하면 새롭게 연결된 것에 한해서만 설정이 적용이 됩니다.

딜레이 제거 임계값: 읽기 전용 인스턴스가 원본 인스턴스의 데이터를 동기화할 때 허용되는 최대 지연 시간을 지정합니다. 읽기 전용 인스턴스의 지연이 이 임계값을 초과하면 가중치에 관계없이 읽기 요청이 인스턴스로 전달되지 않습니다. 값은 1 이상의 정수여야 합니다.

만약 읽기 전용 인스턴스 딜레이가 임계값을 초과할 경우 삭제되며, 삭제된 인스턴스의 가중치는 0으로 자동 설정되고 시스템은 사용자에게 알람을 발송합니다(먼저 '데이터베이스 프록시 마운트 노드 제거' 알람을 구독해야 하며, 설정에 관하여는 알람 정책을 참고 바랍니다.)

읽기 전용 인스턴스의 딜레이가 임계값보다 작을 경우 다시 새롭게 데이터베이스 프록시에 추가됩니다. 동시에 딜레이 제거 기능 활성화 여부에 상관 없이 읽기 전용 인스턴스의 장애가 제거 되어 복구 되면, 복구된 인스턴스도 다시 새롭게 데이터베이스 프록시에 추가됩니다.

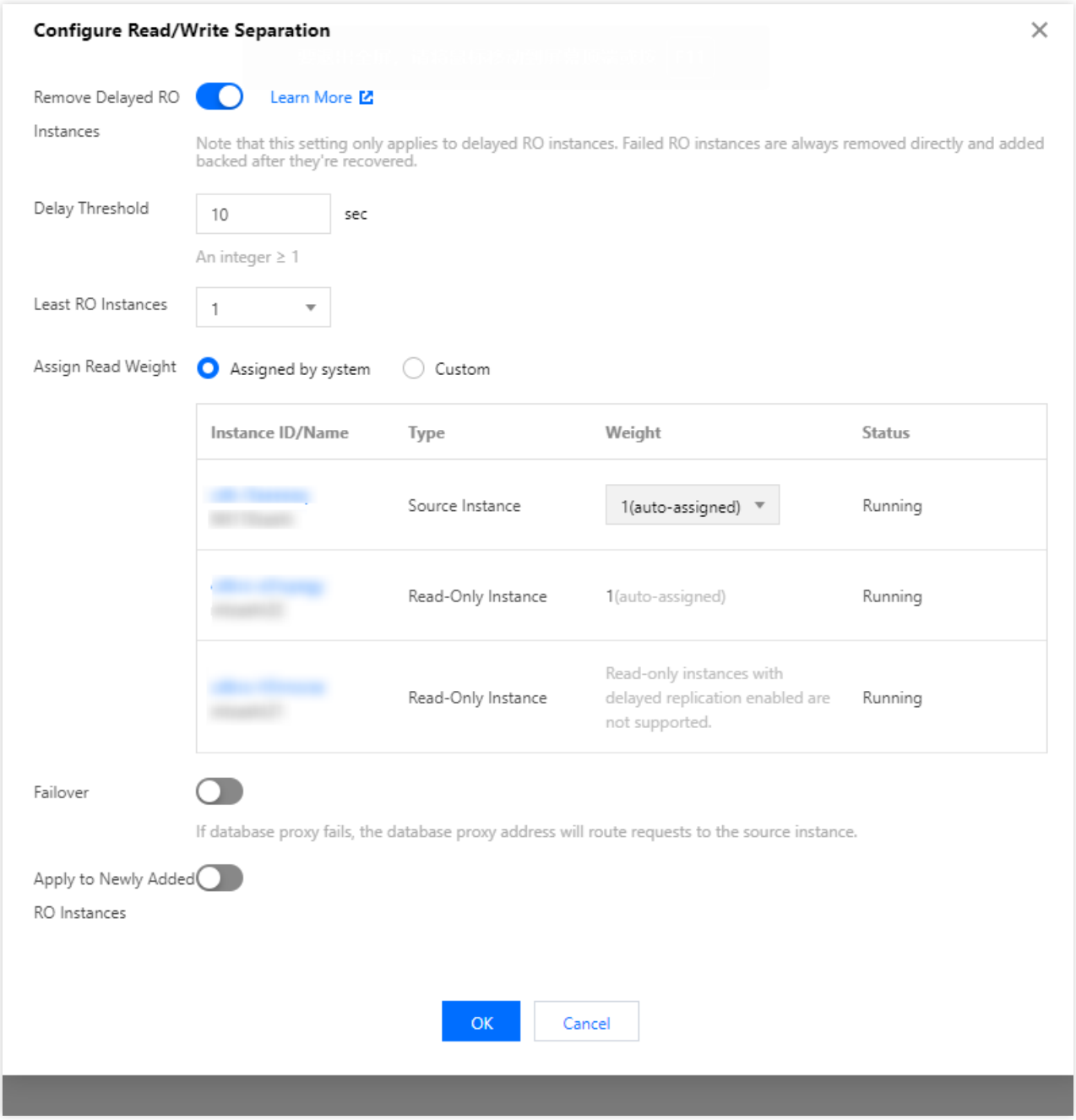
읽기 전용 최소 보관 수량 설정을 통해 데이터베이스 프록시가 현재 라우팅 중인 읽기 전용 인스턴스가 설정한 값보다 작은 것을 발견 시, 오류가 난 읽기 전용 인스턴스를 읽기 전용 인스턴스 개수가 최소 보관 수량에 만족할 때까지 읽기/쓰기 분리로 이동시킵니다.

주의 :

읽기 전용 라이브러리에서 치명적인 장애(예: 시스템 다운)가 발생할 경우, 최소 보류 수량은 치명적인 장애가 발생한 인스턴스에 대하여 작용하지 못합니다.

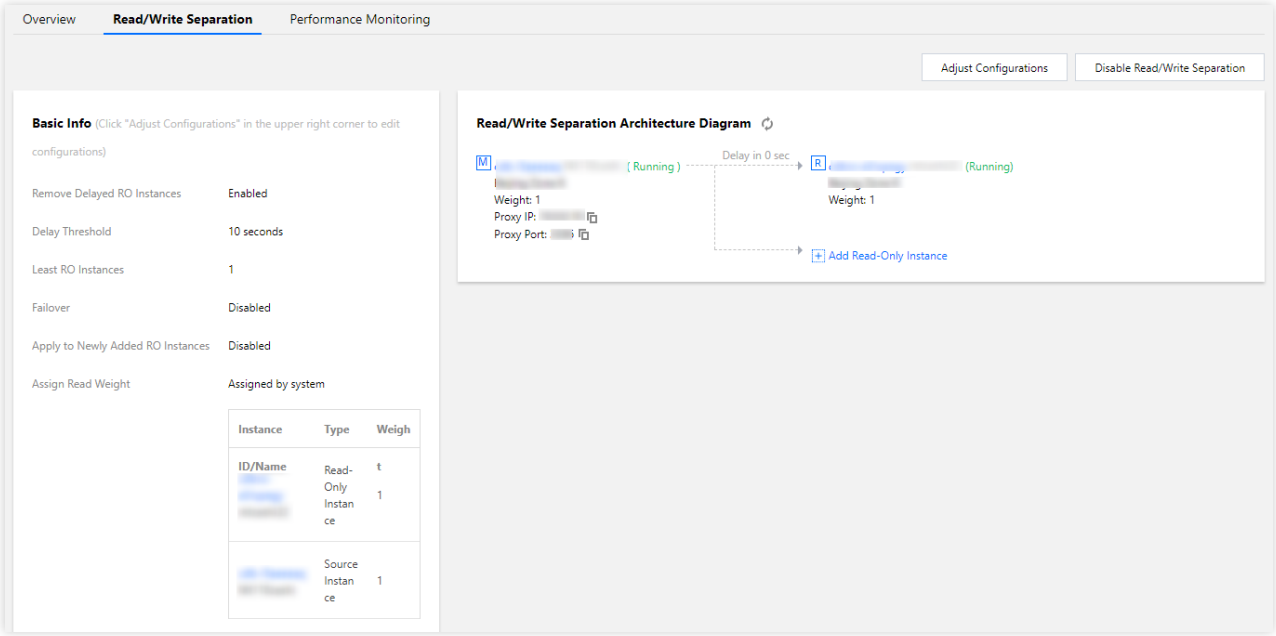
읽기 전용 인스턴스 최소 보관 수: 보장되어야 하는 인스턴스 하한치로, 현재 읽기 전용 인스턴스 수가 해당 하한치와 같거나 적고, 딜레이 시간이 임계값을 초과하면 현재의 읽기 전용 인스턴스는 제거되지 않습니다.

읽기 가중치 할당: 인스턴스에 읽기 가중치를 할당하며, 시스템에서 자동으로 할당하거나 사용자 정의를 선택할 수 있으며 가중치 할당 범위는 0 - 100까지의 정수입니다. 가중치 할당 설정 읽기 후 모든 연결에 즉시 적용됩니다.
데이터베이스 프록시는 가중치 설정에 따라 읽기 요청의 트래픽을 할당합니다. 예를 들어, 2개의 읽기 전용 데이터베이스의 가중치가 각각 10과 20인 경우, 읽기 요청 트래픽은 1:2의 비율로 할당됩니다.
여기서 가중치는 쓰기 요청이 가중치 계산에 참여하지 않고 원본 데이터베이스로 직접 라우팅되기 때문에 읽기 요청 가중치만을 나타냅니다. 예를 들어 클라이언트가 10개의 쓰기 명령문과 10개의 읽기 명령문을 보내고 원본 및 읽기 전용 인스턴스 가중치의 비율이 1:1인 경우 원본 인스턴스는 10개의 쓰기 명령문과 5개의 읽기 명령문을 수신하고 읽기 전용 인스턴스는 5개의 읽기 문만 받습니다.
시스템에 의해 할당을 선택하면 시스템이 인스턴스의 CPU 및 메모리 사양에 따라 자동으로 가중치를 할당하며 이 경우 원본 인스턴스의 가중치만 설정할 수 있습니다.
읽기 전용 인스턴스의 가중치가 0인 경우 데이터베이스 프록시는 인스턴스에 연결하지 않습니다. 가중치가 0에서 다른 값으로 변경되면 가중치는 새 연결에만 적용됩니다.
장애 조치: 이 매개변수를 활성화하거나 비활성화합니다. 모든 읽기 전용 인스턴스가 비정상인 경우 데이터베이스 프록시가 원본 인스턴스에 읽기 요청을 보낼 수 있도록 이 매개변수를 활성화하는 것이 좋습니다.
설명：
장애 조치 설정 후 새로운 연결에 한해서만 설정 적용됩니다.
읽기 전용 인스턴스 자동 추가: 활성화 여부를 설정합니다. 활성화 후 새로운 읽기 전용 인스턴스를 구매하면 데이터베이스 프록시에 자동으로 추가됩니다.
시스템에서 자동으로 읽기 가중치를 할당할 경우 구매한 읽기 전용 인스턴스에 규격 크기에 따라 기본 가중치를 할당합니다.
읽기 가중치가 사용자 정의인 경우 구매한 읽기 전용 인스턴스의 가중치는 기본적으로 0이며, 데이터베이스 프록시 읽기/쓰기 분리의 설정 변경을 통해 수정할 수 있습니다.



데이터베이스 프록시 읽기/쓰기 분리 사용

데이터베이스 프록시 읽기/쓰기 분리 기능이 성공적으로 활성화된 후 데이터베이스 프록시 페이지는 다음과 같습니다.



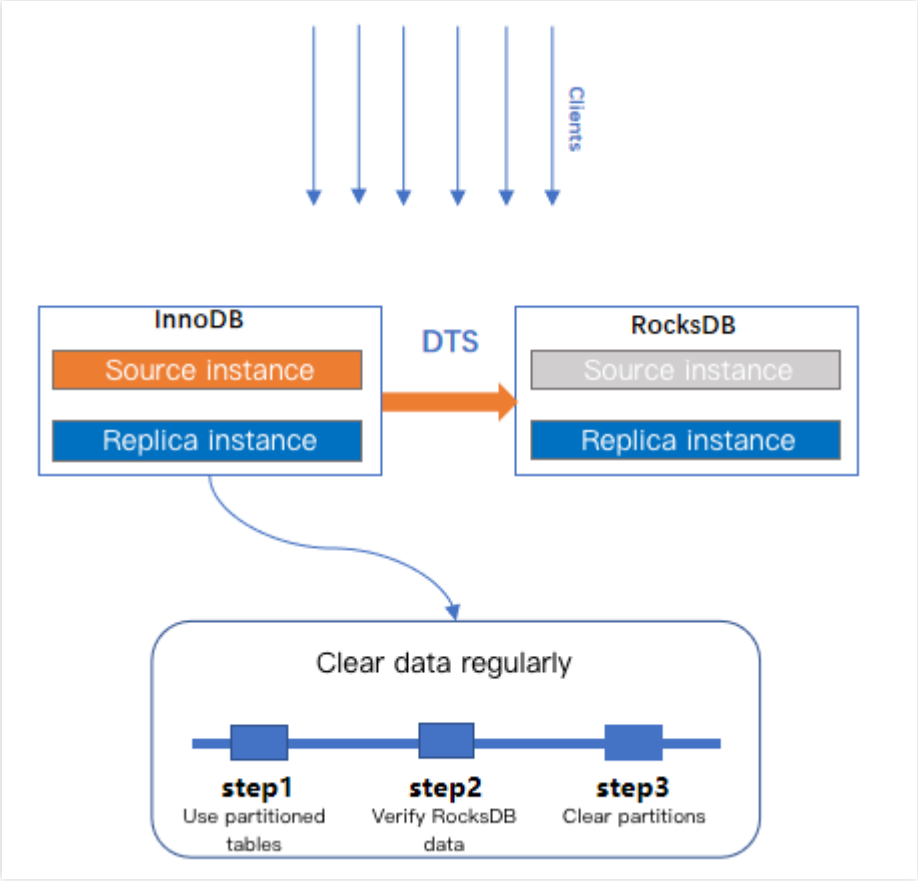
관련 문서

데이터베이스 프록시 연결 주소 설정
데이터베이스 프록시 네트워크 전환
세션 레벨 연결 풀 설정

DTS를 사용하여 InnoDB에서 RocksDB로 데이터 마이그레이션

최종 업데이트 날짜: : 2024-07-25 16:38:48

TXRocks는 매우 인기 있는 고성능 KV(key-value) 저장소인 RocksDB를 기반으로 Tencent의 TXSQL 팀에서 개발한 트랜잭션 스토리지 엔진입니다.
InnoDB에서 사용하는 B+Tree 구조와 비교하여 TXRocks에서 채택한 LSM Tree 구조는 훨씬 적은 저장 공간을 사용합니다. InnoDB의 B+Tree 분할은 종종 half-full 페이지, 유휴 페이지 및 공간 낭비를 초래합니다. 따라서 InnoDB는 유효 페이지 활용도가 낮습니다. TXRocks SST 파일의 크기는 일반적으로 수십 또는 수백 MB 이상의 값으로 설정됩니다. 따라서 TXRocks는 4K 정렬로 인한 낭비가 훨씬 적습니다. SST 파일은 Block으로 나누어져 있지만 해당 Block을 정렬할 필요는 없습니다. 또한 TXRocks SST 파일은 접두사 압축을 사용하므로 동일한 접두사가 있는 데이터 레코드에 대해 하나의 레코드만 생성됩니다. 다른 레벨의 SST 파일은 다른 압축 알고리즘을 채택하여 저장 공간 오버헤드를 더욱 줄일 수 있습니다. 일반적으로 저장 공간 사용량을 50%까지 줄일 수 있습니다.
InnoDB 데이터를 DTS를 통해 RocksDB에 동기화하여 쓰기 성능을 개선하고 저장 공간을 절약할 수 있습니다.



설명：
DTS(Data Transmission Service)는 데이터 마이그레이션, 동기화 및 구독과 같은 기능을 통합하여 업무 중단 없이 데이터베이스를 마이그레이션하고 실시간 동기화 채널을 통해 원격 재해 복구를 위한 고가용성 데이터베이스 아키텍

처를 구축할 수 있도록 지원하는 데이터 전송 서비스입니다. 데이터 구독 기능을 사용하면 TencentDB 인스턴스에서 점진적으로 업데이트되는 데이터에 실시간으로 액세스할 수 있으므로 귀하의 비즈니스 요구 사항에 따라 이러한 데이터를 사용할 수 있습니다.

주의 사항

데이터 동기화를 위해 DTS를 사용할 때 증분 데이터가 RocksDB에 동기화되고 Delete 작업이 차단되는지 확인하십시오.
RocksDB로 데이터를 전송한 후 데이터 유효성을 먼저 확인한 다음 원본 데이터베이스의 데이터를 지워 저장 공간 사용량을 줄입니다.
원본 테이블에 파티션된 테이블을 사용하여 데이터 지우기 효율성을 높일 수 있습니다(파티션된 테이블의 사용 제한에 주의하고 타깃 테이블은 파티셔닝을 지원하지 않습니다).
RocksDB는 저장 공간 사용량을 효과적으로 줄이기 위해 정기적으로 compaction 작업을 수행합니다.
RocksDB 엔진에 대한 자세한 내용은 Instructions를 참고하십시오.

작업 단계

1. DTS 콘솔에 로그인하고 왼쪽 사이드바에서 **데이터 마이그레이션**을 선택한 다음 **마이그레이션 작업 생성**을 클릭하여 마이그레이션 작업 생성 페이지로 이동합니다.
2. 마이그레이션 작업 생성 페이지에서 원본 및 타깃 인스턴스의 유형, 리전 및 사양을 선택하고 **즉시 구매**를 클릭합니다.

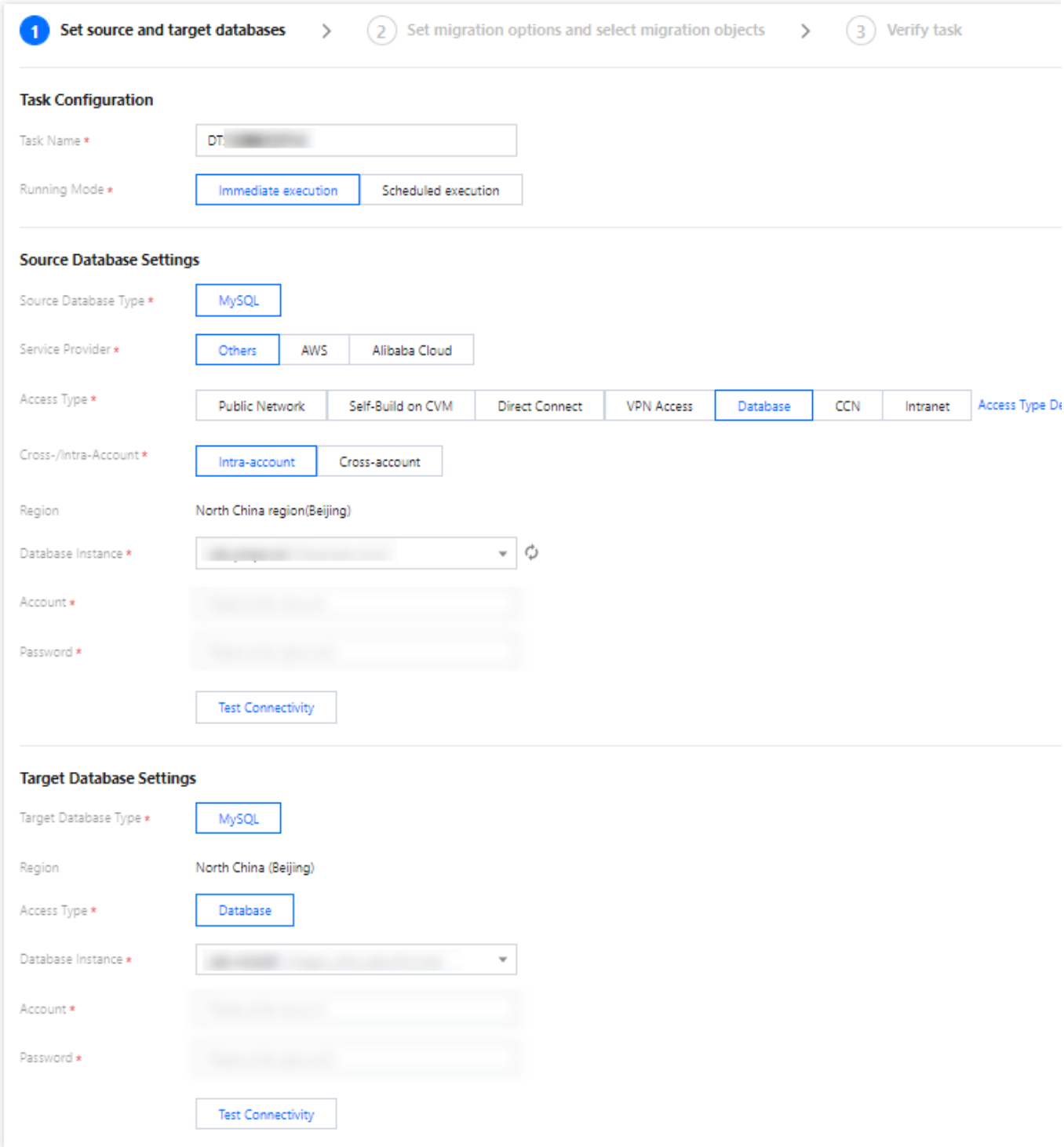
설정 항목	설명
원본 인스턴스 유형	원본 데이터베이스 유형을 선택합니다. 구매 후 변경할 수 없습니다. 여기에서는 'MySQL'을 선택합니다.
원본 인스턴스 리전	원본 데이터베이스 리전을 선택합니다. 원본 데이터베이스가 자체 구축된 데이터베이스인 경우 가장 가까운 리전을 선택하십시오.
타깃 인스턴스 유형	타깃 데이터베이스 유형을 선택합니다. 구매 후 변경할 수 없습니다. 여기에서 'MySQL'을 선택합니다.
타깃 인스턴스 리전	타깃 데이터베이스 리전을 선택합니다.
사양	귀하의 업무 여건에 따라 마이그레이션 연계 사양을 선택합니다. 다양한 사양의 성능 및 결제 세부 정보는 과금 개요를 참고하십시오.

3. 마이그레이션 작업의 **작업** 열에서 **더 보기 > 설정**을 선택합니다. 원본 및 타깃 데이터베이스 설정 페이지에서 작업, 원본 데이터베이스 및 타깃 데이터베이스 설정을 구성합니다. 원본 및 타깃 데이터베이스가 연결 테스트를 통과

한 후 **저장**을 클릭합니다.

설명：

연결 테스트에 실패하면 DB 연결 확인의 지침에 따라 문제를 해결하고 수정한 후 다시 시도하십시오.



4. 마이그레이션 옵션 설정 및 마이그레이션 객체 선택 페이지에서 마이그레이션 유형과 마이그레이션 객체를 설정하고 **저장**을 클릭합니다.

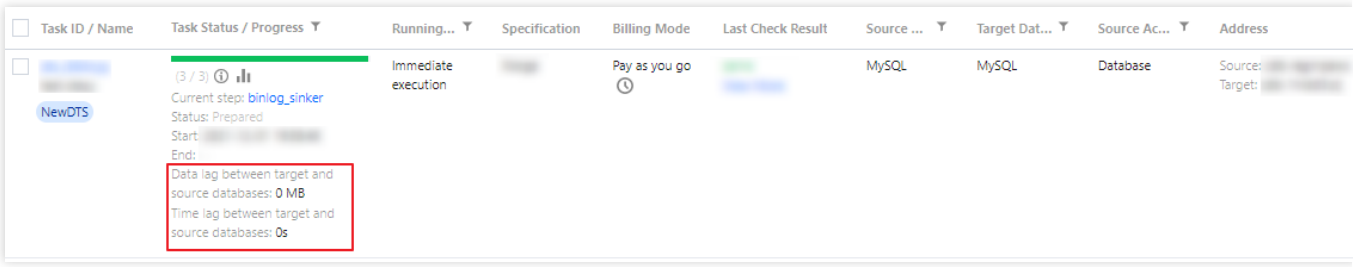
설명：

마이그레이션 중에 테이블에 Online DDL 작업을 수행하기 위해 gh-ost 및 pt-osc와 같은 도구를 사용하려면, **마이그레이션 객체**로 테이블만 선택하는 것이 아니라 테이블이 있는 전체 데이터베이스(또는 전체 인스턴스)를 선택해야

합니다. 그렇지 않으면 Online DDL 변경으로 생성된 임시 테이블 데이터를 대상 데이터베이스로 마이그레이션할 수 없습니다.
마이그레이션하는 동안 테이블 이름을 rename하려면(예: table A를 table B로 rename) **마이그레이션 객체**로 table A만 선택하는 것이 아니라 table A가 있는 전체 데이터베이스(또는 전체 인스턴스)를 선택해야 합니다. 그렇지 않으면 시스템에서 오류를 보고합니다.

설정 항목	설명
마이그레이션 유형	시나리오에 따라 유형을 선택하십시오. 구조적 마이그레이션: 원본 데이터베이스의 데이터베이스 및 테이블과 같은 구조적 데이터가 마이그레이션됩니다. 전체 마이그레이션: 전체 데이터베이스가 마이그레이션됩니다. 마이그레이션된 데이터는 작업이 시작될 때만 원본 데이터베이스의 기존 콘텐츠가 되지만 작업이 시작된 후 원본 데이터베이스에 기록된 증분 데이터는 포함되지 않습니다. 전체 + 증분 마이그레이션: 마이그레이션된 데이터에는 작업이 시작될 때 원본 데이터베이스의 기존 콘텐츠와 작업이 시작된 후 원본 데이터베이스에 기록된 증분 데이터가 포함됩니다. 마이그레이션 중에 원본 데이터베이스에 데이터 쓰기가 있고 논스톱 방식으로 원활하게 데이터를 마이그레이션하려면 이 옵션을 선택하십시오.
객체 마이그레이션	전체 인스턴스: information_schema, mysql, performance_schema 및 sys와 같은 시스템 데이터베이스를 제외한 전체 데이터베이스 인스턴스를 마이그레이션합니다. 지정된 객체: 지정된 객체를 마이그레이션합니다.
객체 지정	원본 데이터베이스 객체에서 마이그레이션할 객체를 선택하고 선택한 객체 상자로 이동합니다.
계정 마이그레이션	원본 데이터베이스의 계정 정보를 마이그레이션하려면 이 기능을 선택하십시오.

5. 확인 작업 페이지에서 확인을 진행하고 확인 통과 후 **작업 실행**을 클릭합니다.
6. 데이터 마이그레이션 작업 목록으로 돌아가면 작업이 준비 중 상태로 들어간 것을 볼 수 있습니다. 1 - 2분 후 데이터 마이그레이션 작업이 시작됩니다.
구조적 마이그레이션 또는 **전체 마이그레이션**을 선택: 완료되면 작업이 자동 중지됩니다.
전체 + 증분 마이그레이션 선택: 전체 마이그레이션이 완료된 후 마이그레이션 작업은 자동으로 중지되지 않는 증분 데이터 동기화 단계에 자동으로 들어갑니다. 증분 데이터 동기화를 수동으로 중지하려면 **완료**를 클릭해야 합니다. 적절한 시기에 증분 데이터 동기화 및 비즈니스 전환을 수동으로 완료합니다.
마이그레이션 작업이 증분 동기화 단계에 있고 지연 상태가 아닌지 확인합니다. 그렇다면 몇 분 동안 원본 데이터베이스에 데이터 쓰기를 중지하십시오.
타깃 데이터베이스와 원본 데이터베이스 간의 데이터 간격이 0MB이고 둘 사이의 시간 지연이 0초인 경우 수동으로 증분 동기화를 완료합니다.



7. 마이그레이션 작업 상태가 **작업 성공**이 된 후 InnoDB 데이터가 RocksDB에 동기화됩니다.

웹 애플리케이션을 위한 LAMP 스택 구축

최종 업데이트 날짜: : 2024-07-25 16:38:48

LAMP는 Linux+Apache+Mysql/MariaDB+Perl/PHP/Python을 가리키며, 동적 웹사이트 또는 서버를 구축하는 데 사용되는 오픈 소스 소프트웨어입니다. 프로그램은 각각 독립적이지만, 항상 함께 사용되기 때문에 상호 호환성이 점점 증가하고 있습니다. 또한 공동으로 강력한 Web 응용 프로그램 플랫폼을 구성합니다.
본 튜토리얼은 사용자가 다음 과정을 완료할 수 있도록 안내합니다: 1개의 Tencent CDB 인스턴스를 실행하고, Tencent CVM을 통해 1개의 LAMP 응용 프로그램을 설정하여, Tencent CDB 인스턴스의 고가용성 환경을 연결하는 데 사용합니다.
Tencent CDB 인스턴스를 실행하면 데이터베이스와 환경의 라이프사이클을 각각 분리합니다. 이를 통해 사용자가 여러 서버에서 동일한 데이터베이스로 연결할 수 있도록 합니다. 또한 사용자가 데이터베이스 설치, 배포, 버전 업데이트 및 장애 처리 등의 문제에 더 이상 신경 쓰지 않을 수 있도록 데이터베이스의 유지보수를 간소화합니다.
설명：
본 튜토리얼에서 사용하는 CDB 인스턴스와 CVM 인스턴스는 동일 리전에 속합니다. 사용자의 CDB 인스턴스와 CVM 인스턴스가 다른 리전에 속할 경우, 외부 네트워크 액세스를 참조하십시오.

CDB 인스턴스 초기화

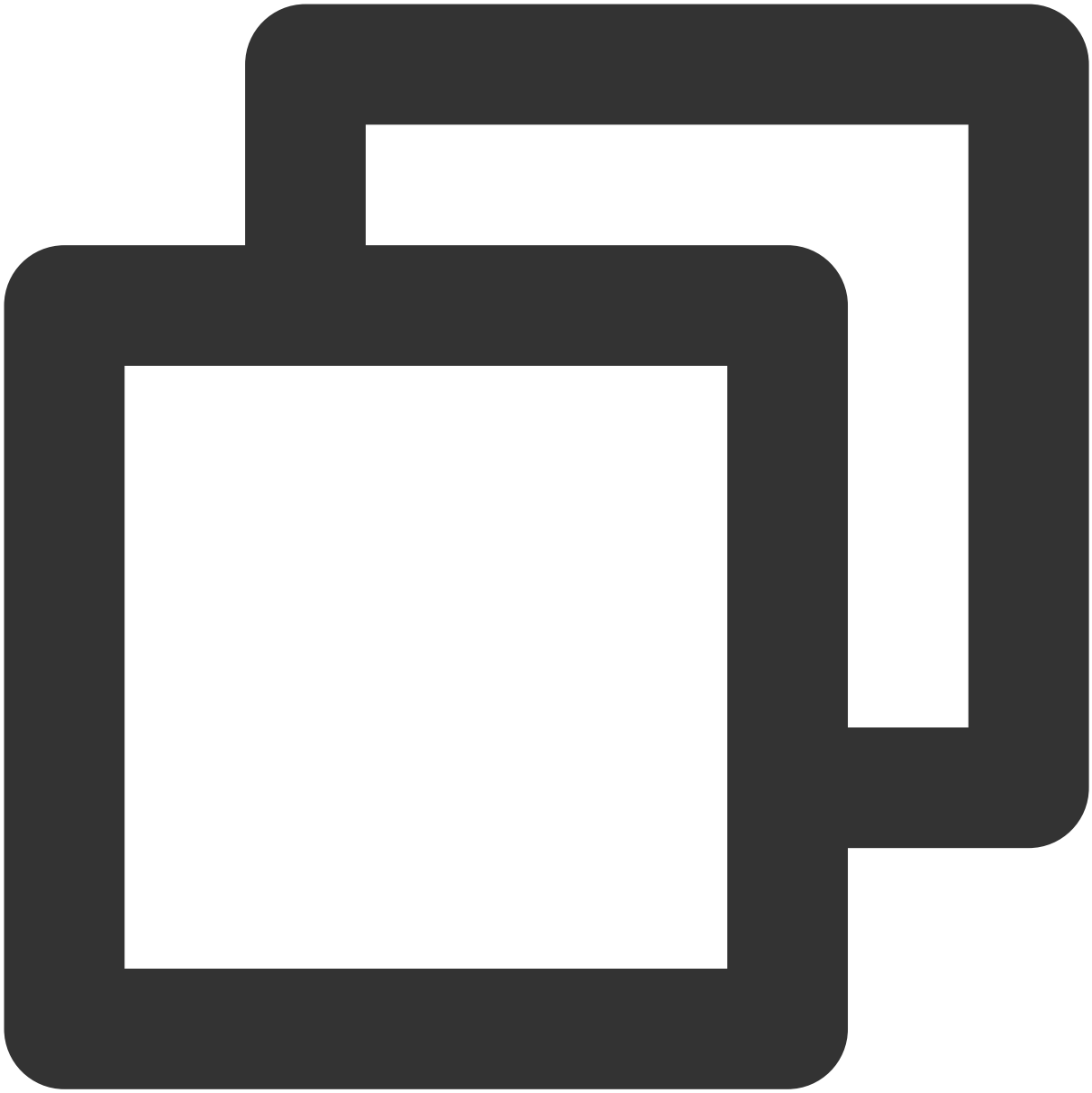
CDB의 구매와 초기화는 구매 방식과 MySQL 데이터베이스 초기화를 참조하십시오.

CVM 인스턴스에 로그인하십시오.

CVM의 구매와 액세스는 Linux CVM 시작하기를 참조하십시오. 본 튜토리얼에서는 CentOS 시스템을 사용합니다.

MySQL 클라이언트 설치

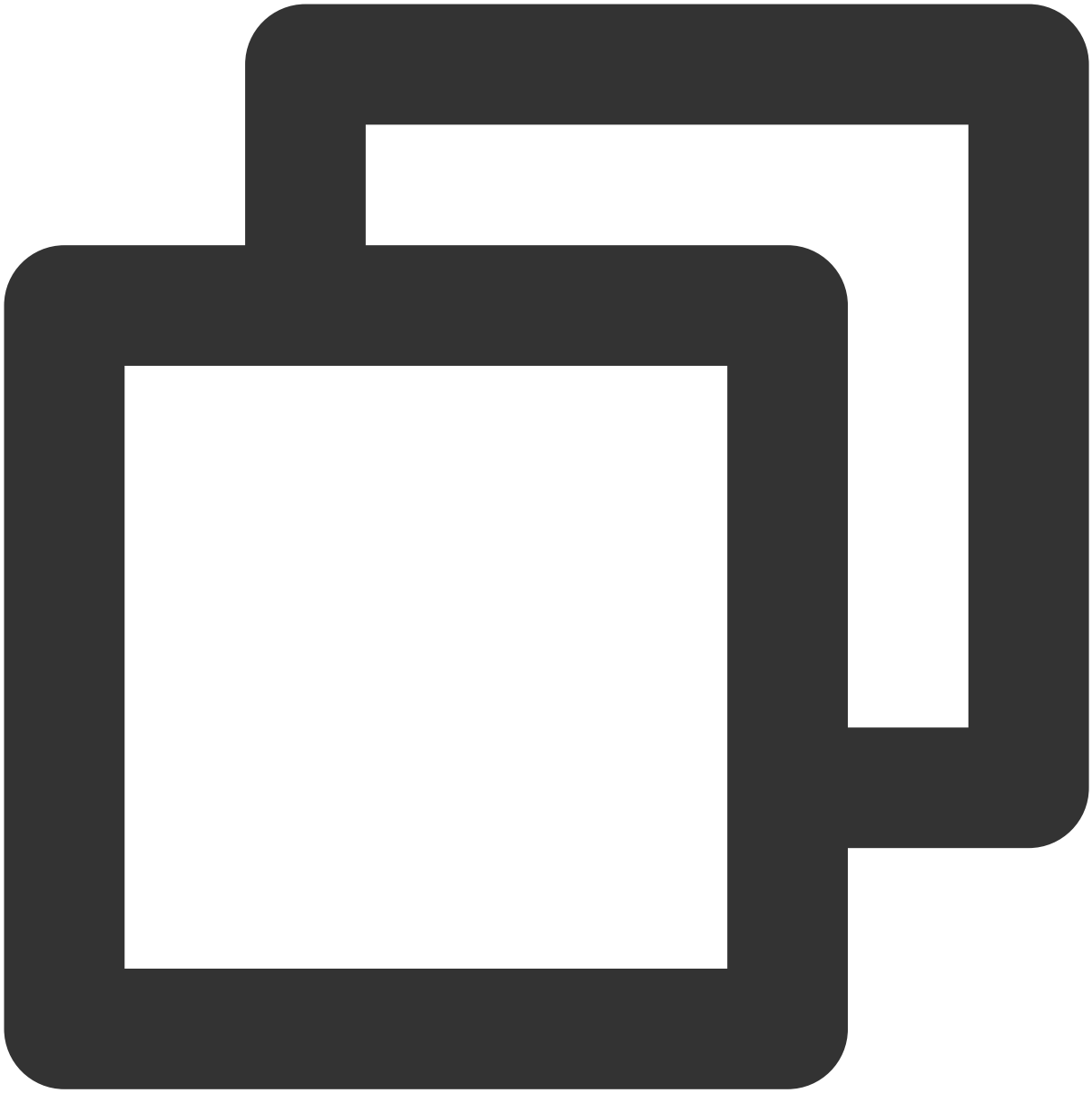
1. CVM 인스턴스에서 `yum` 을 사용해 MySQL 클라이언트를 설치합니다.



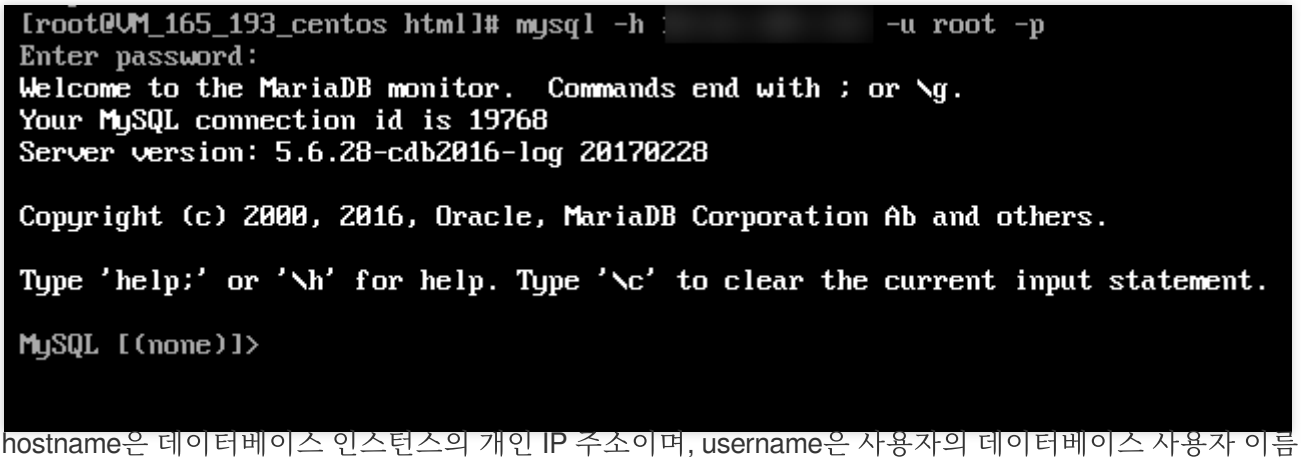
```
yum install mysql -y
```

```
[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. 설치가 완료되면 Tencent CDB 인스턴스에 연결됩니다.

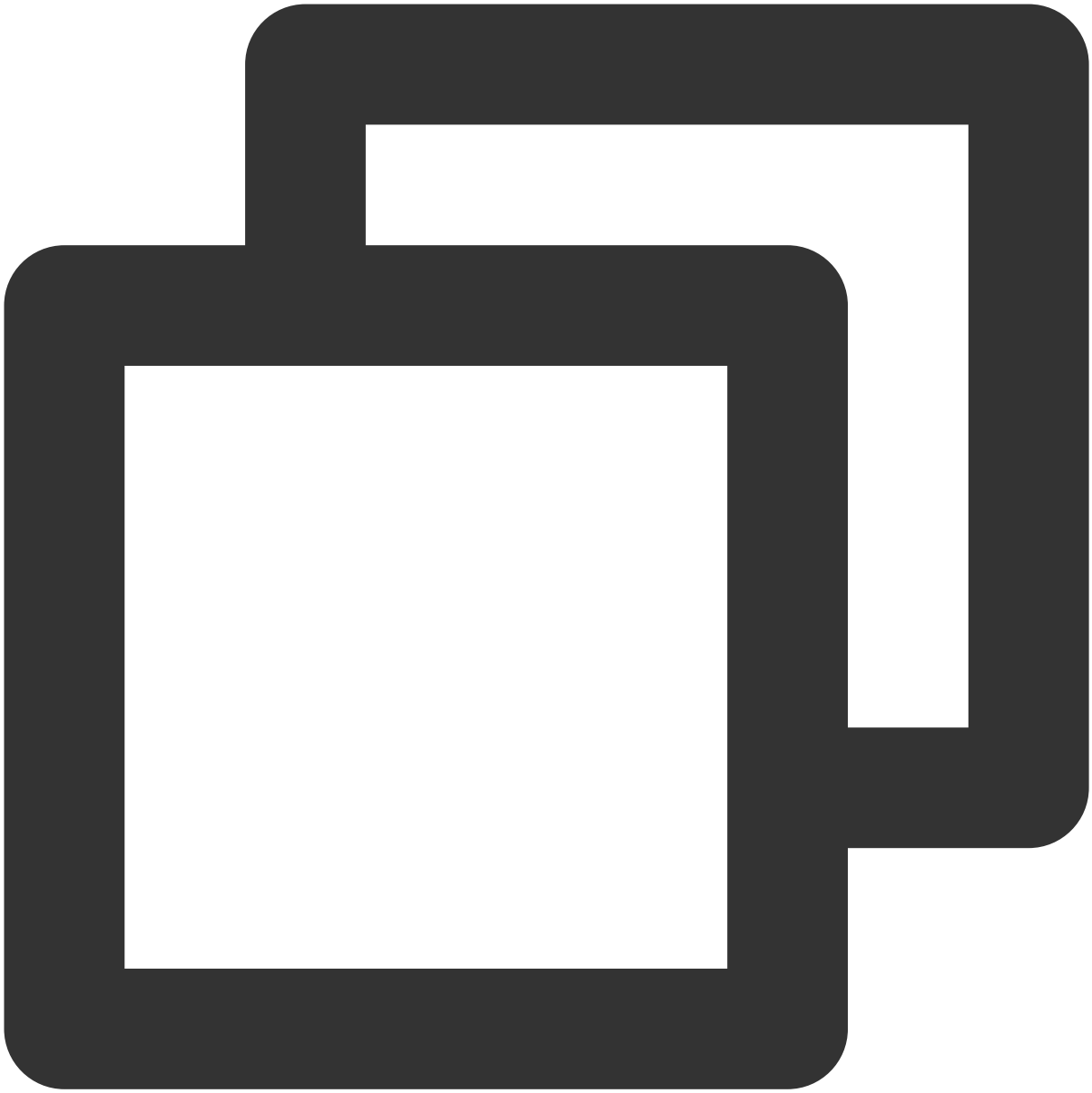


```
mysql -h hostname -u username -p
```



이 중, hostname은 데이터베이스 인스턴스의 개인 IP 주소이며, username은 사용자의 데이터베이스 사용자 이름입니다.

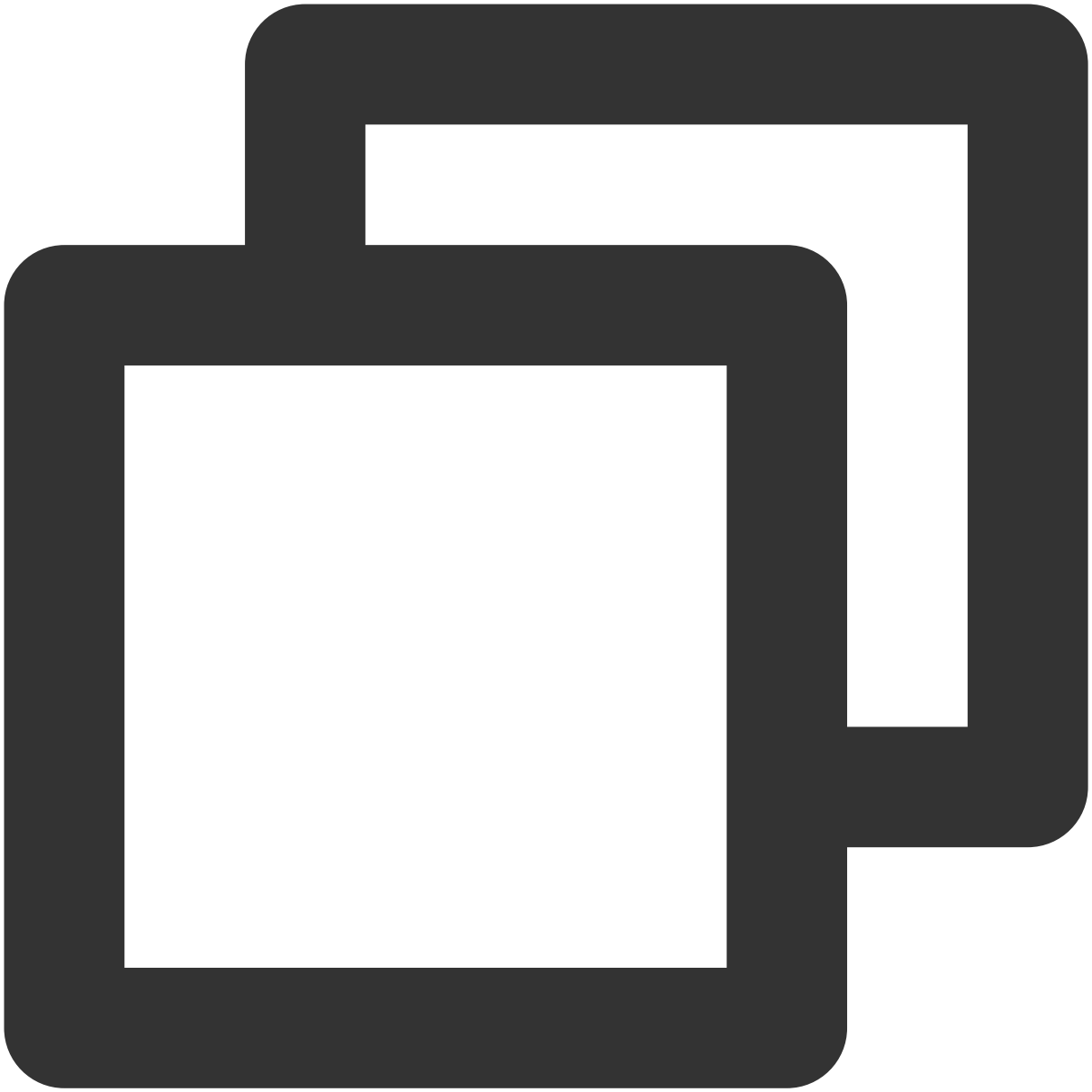
3. 연결 성공 후 즉시 데이터베이스에서 나가 다음 작업을 진행할 수 있습니다.



```
quit;
```

Apache 서비스 설치

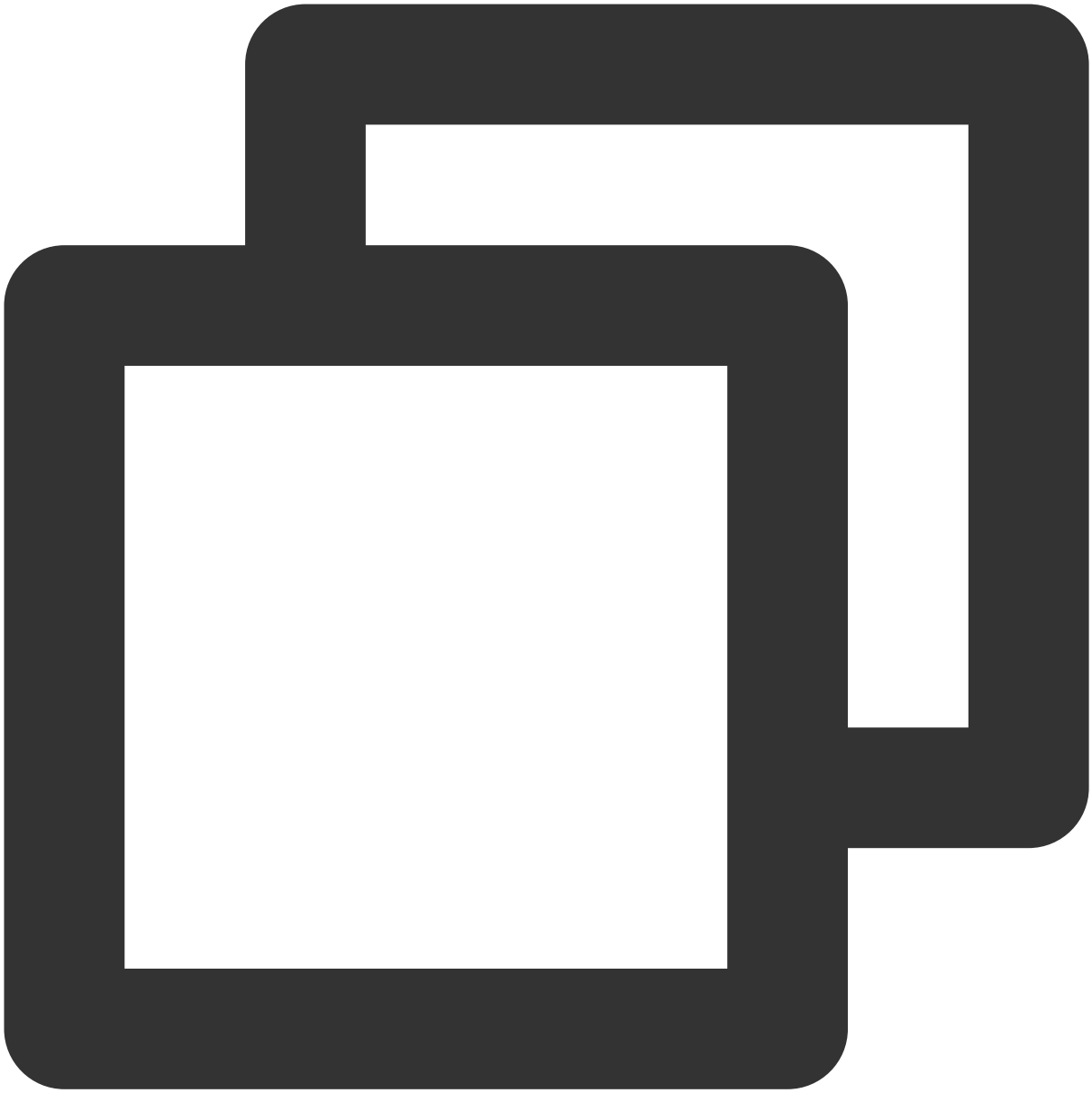
1. CVM 인스턴스에서 `yum` 을 사용해 Apache를 설치합니다.



```
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 서비스를 실행합니다.



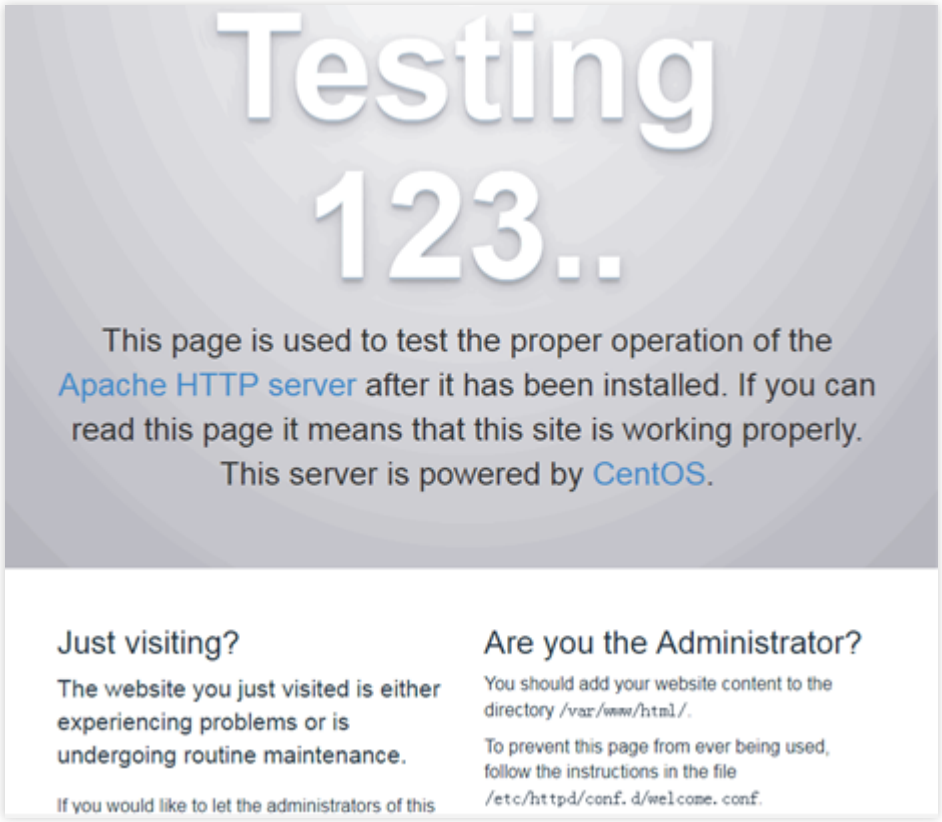
```
service httpd start
```

3. Apache 테스트

주의 :

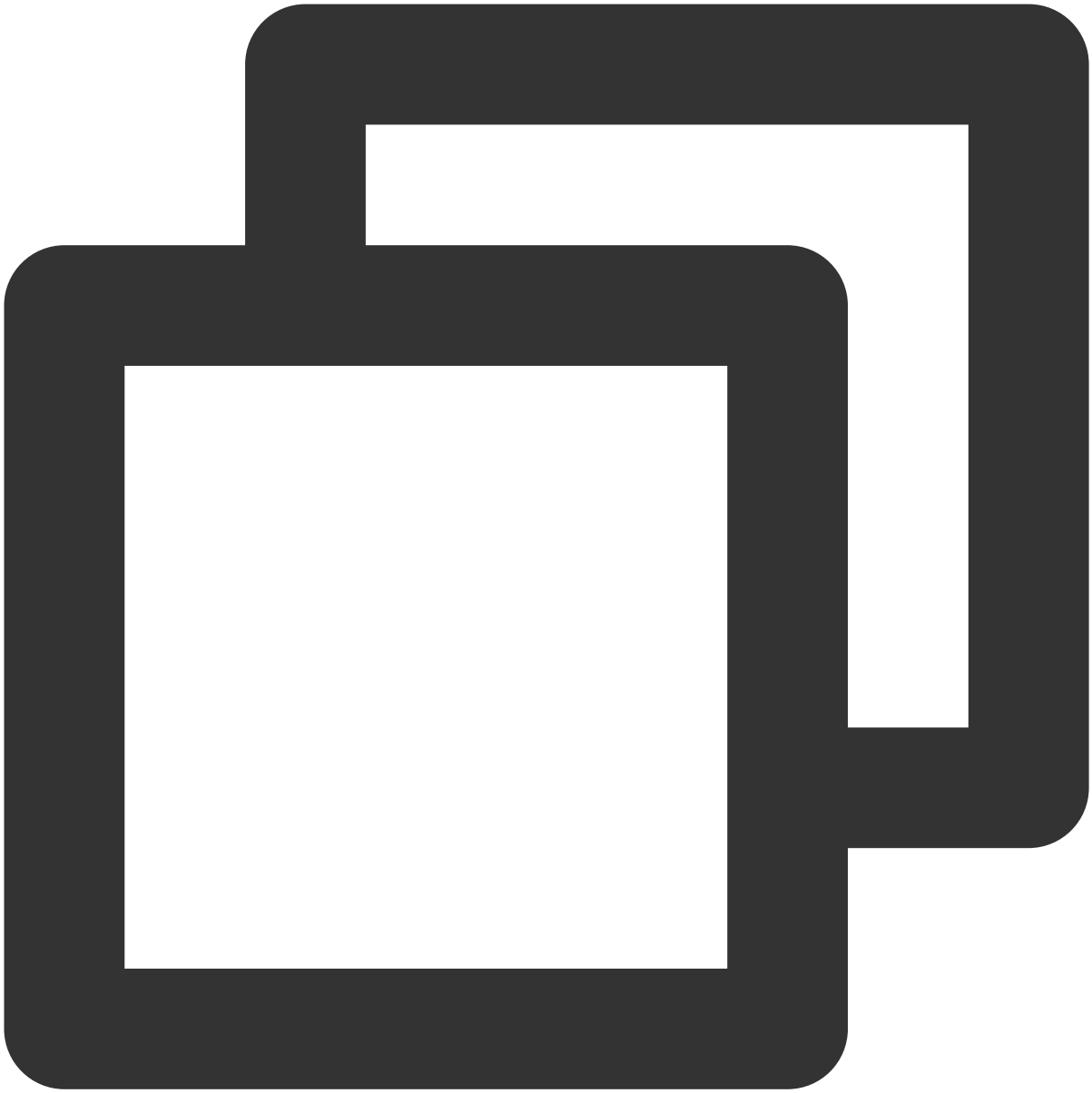
이 단계에서 사용자는 CVM 보안 그룹에서 출처를 **all**로 설정하고, 포트 프로토콜을 **TCP:80**인 Inbound rule로 설정해야 합니다. 보안 그룹 설정은 Security Group을 참조하십시오.

사용자 로컬 브라우저에 `http://xxx.xxx.xxx.xxx/` 를 입력하면(이 중, `xxx.xxx.xxx.xxx` 는 사용자의 CVM 공인 IP 주소입니다), 다음 화면이 나타나면서 Apache가 실행됩니다.



PHP 설치

1. CVM 인스턴스에서 `yum` 을 사용해 PHP를 설치합니다.

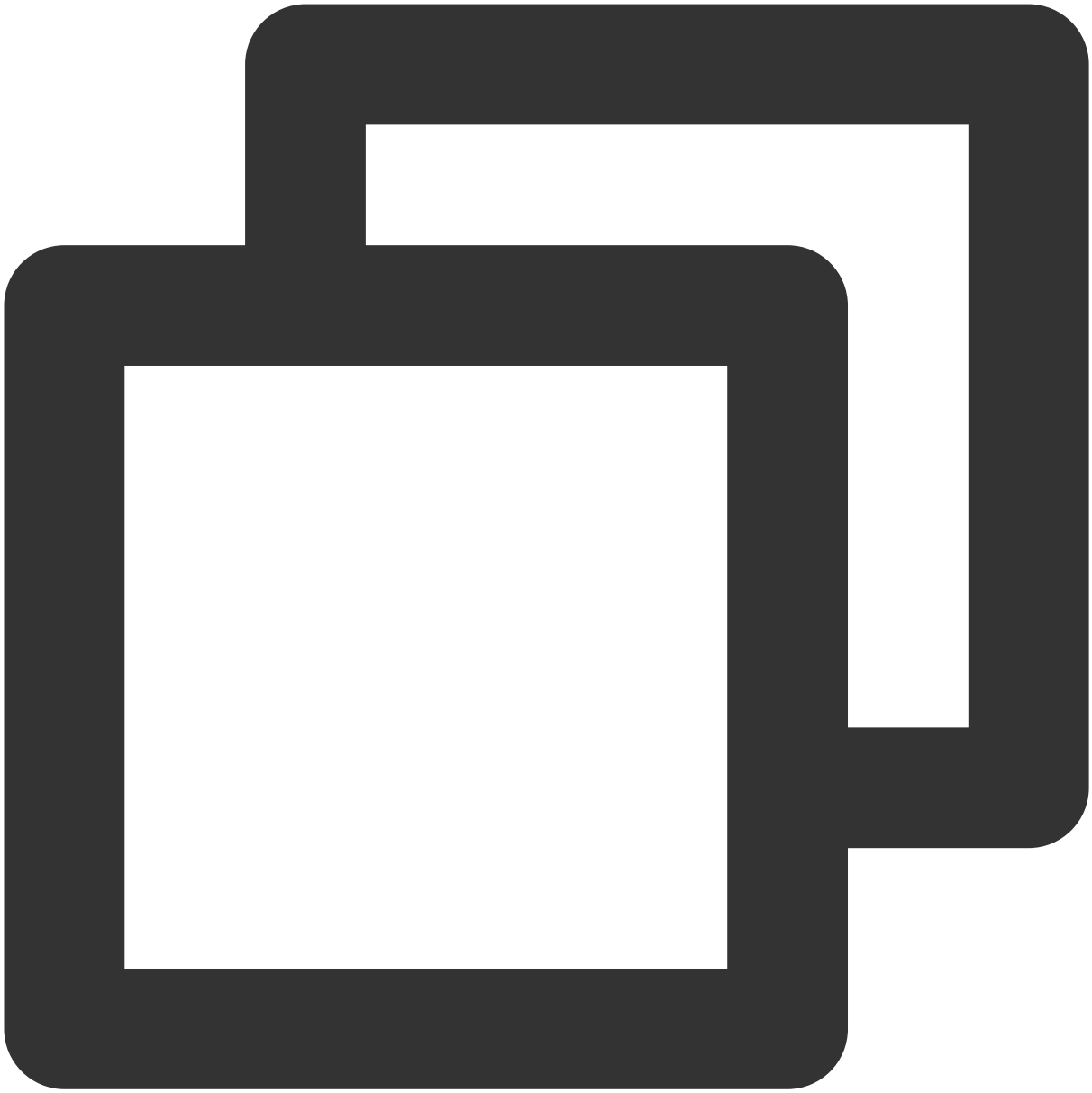


```
yum install php -y
```

```
[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
```

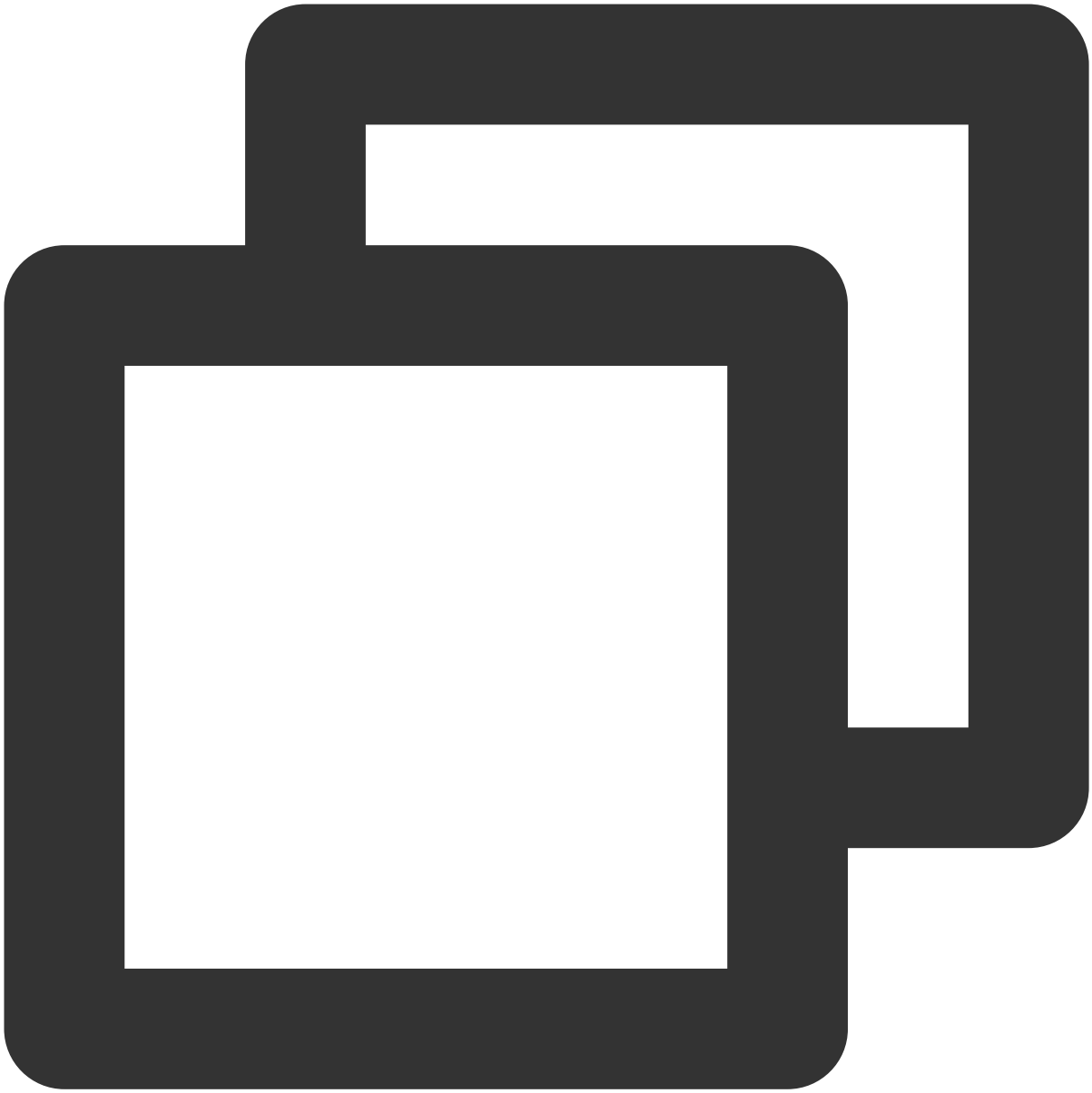
프로젝트 테스트 LAMP 환경 생성

1. 아래의 예시 코드와 같이, CVM의 `/var/www/html` 디렉터리에서 1개의 info.php 파일을 생성합니다.



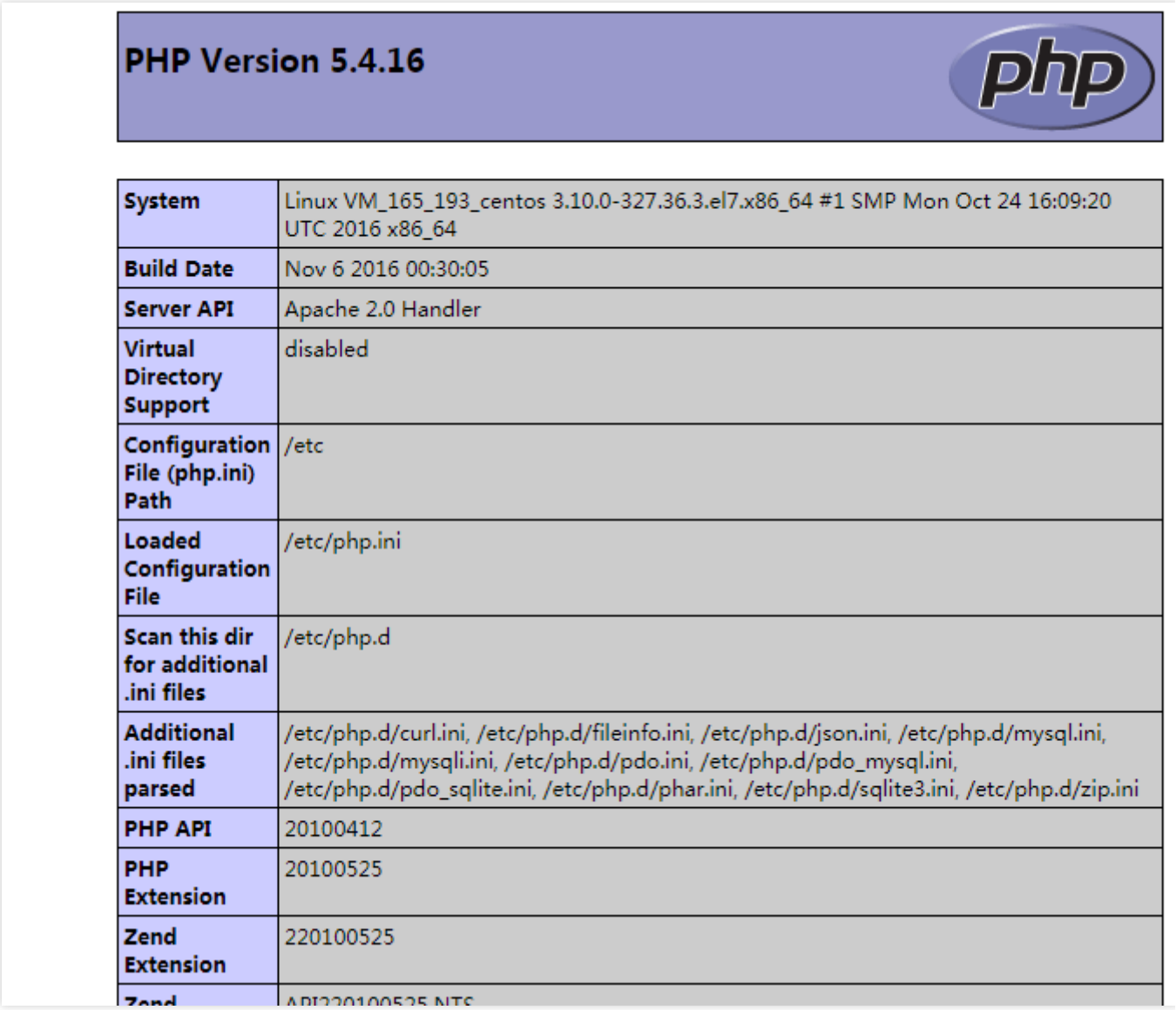
```
<?php phpinfo(); ?>
```

2. Apache 서비스를 재시작합니다.



```
service httpd restart
```

3. 사용자 로컬 브라우저에 `http://xxx.xxx.xxx.xxx/info.php` 를 입력하면(이 중, `xxx.xxx.xxx.xxx` 는 사용자의 CVM 공인 IP 주소입니다), 다음 화면이 나타나면서 LAMP 서비스 배포에 성공합니다.



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 웹사이트 구축

최종 업데이트 날짜: : 2024-07-25 16:38:48

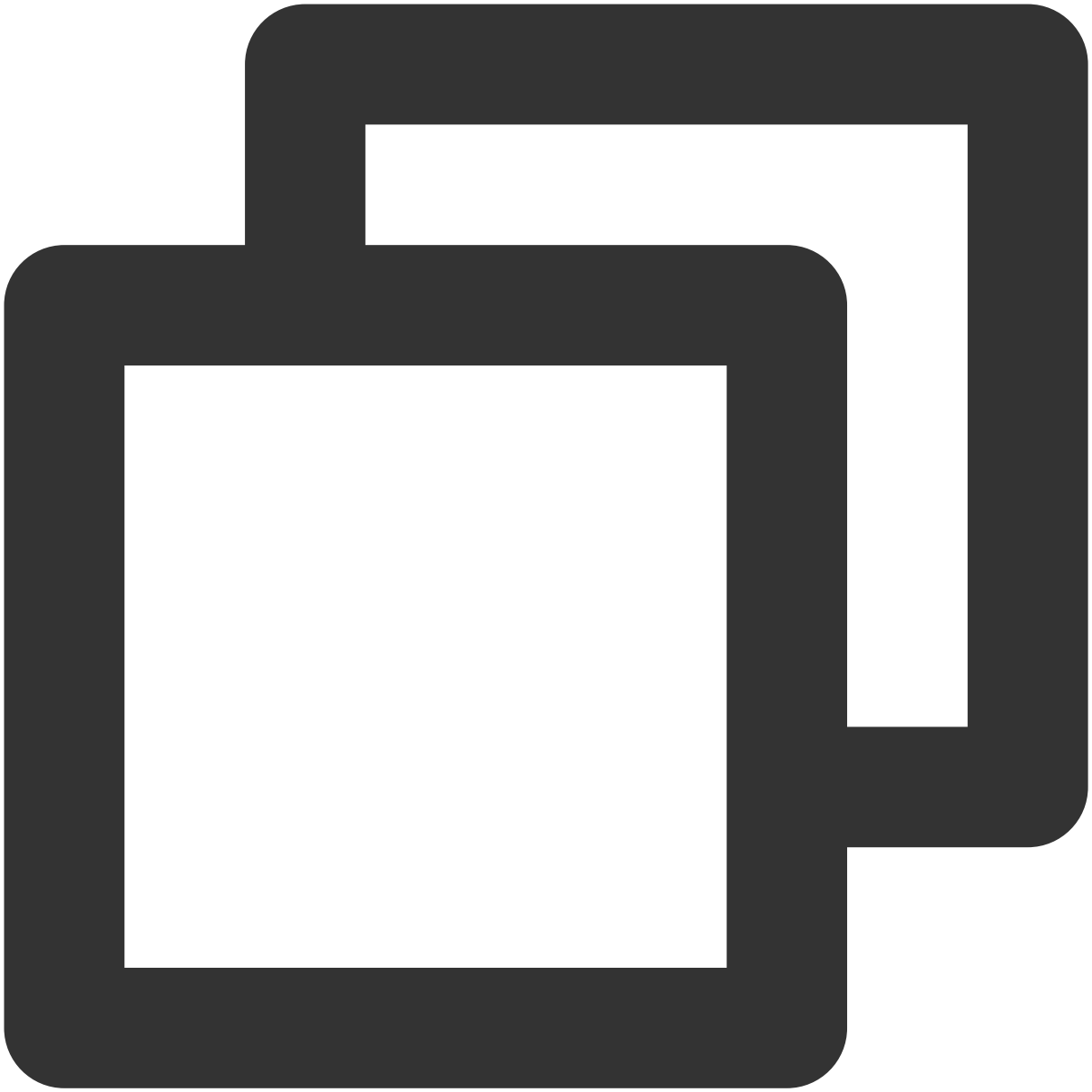
Drupal은 PHP로 작성된 오픈 소스 콘텐츠 관리 프레임워크(Content Management Framework)로, 콘텐츠 관리 시스템(Content Management System)과 PHP 개발 프레임워크(Framework)로 구성되어 있습니다. Drupal은 개인 블로그에서 대규모 커뮤니티에 이르기까지 풍부한 기능을 갖춘 동적 웹 사이트를 구축하는 데 사용할 수 있습니다.
본 튜토리얼은 CVM 인스턴스에서 Drupal 전자상거래 웹사이트를 구축하는 방법을 설명합니다.
사용된 소프트웨어 환경: CentOS7.2, Drupal7.56, PHP5.4.16.

CVM 인스턴스에 로그인

CVM 인스턴스 구매 및 접근 방법에 대한 자세한 내용은 Linux 기반 CVM 시작하기를 참고하십시오.

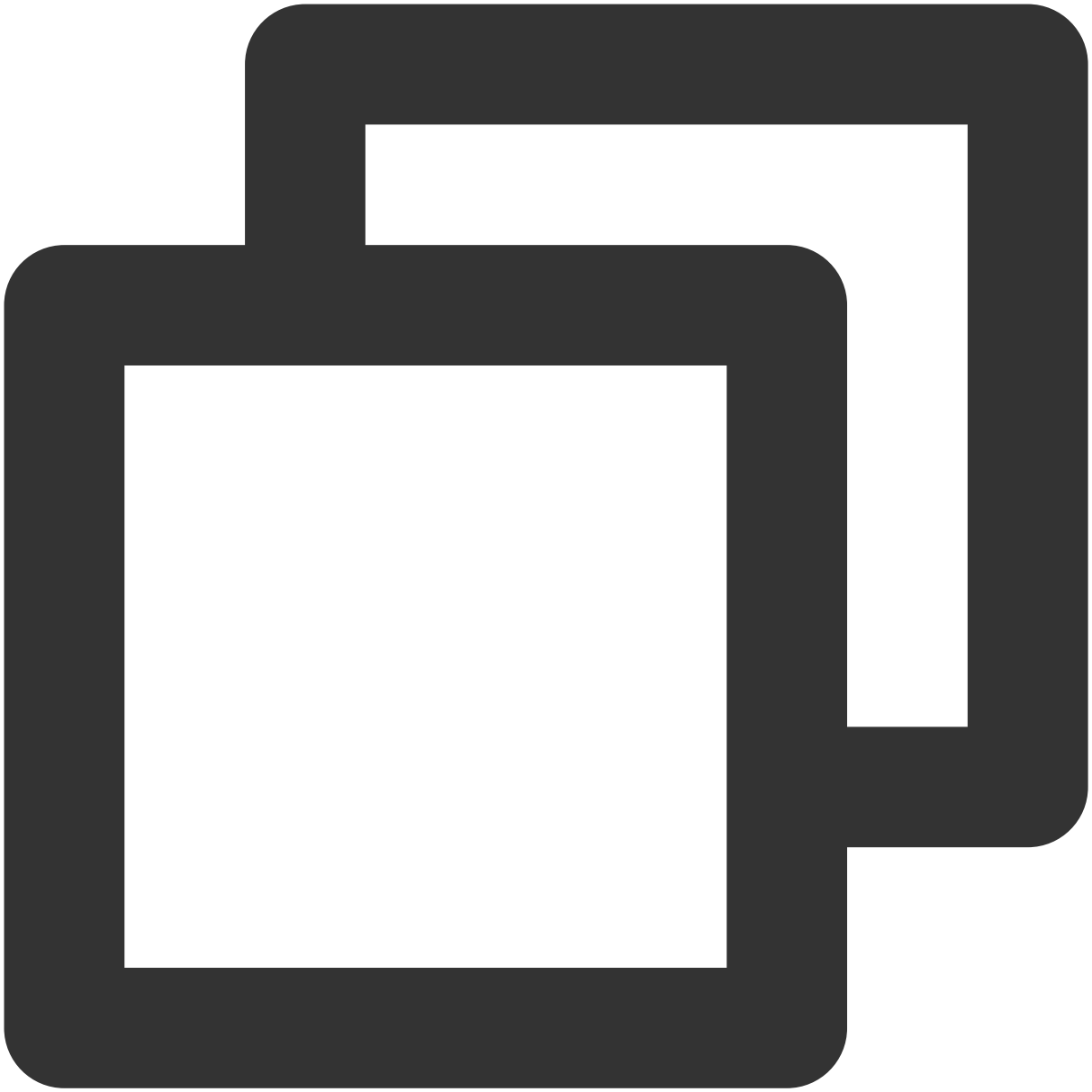
MariaDB 서비스 설치

1. MariaDB는 기본적으로 CentOS v7 이상에서 지원되므로 여기서는 MariaDB를 사용합니다. `yum` 을 사용하여 CVM 인스턴스에 MariaDB 서비스를 설치합니다.



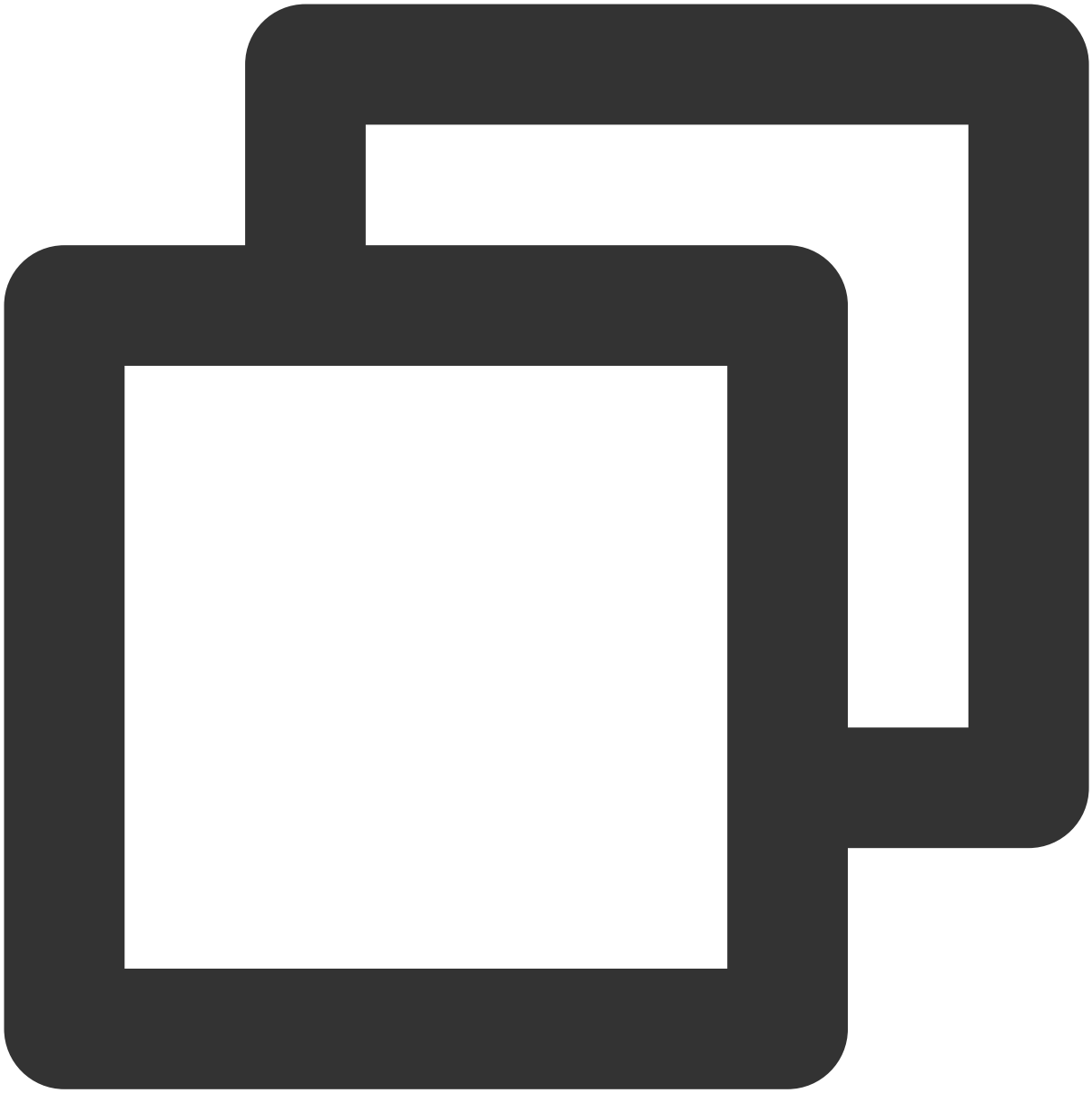
```
yum install mariadb-server mariadb -y
```

2. MariaDB 서비스를 실행합니다.



```
systemctl start mariadb
```

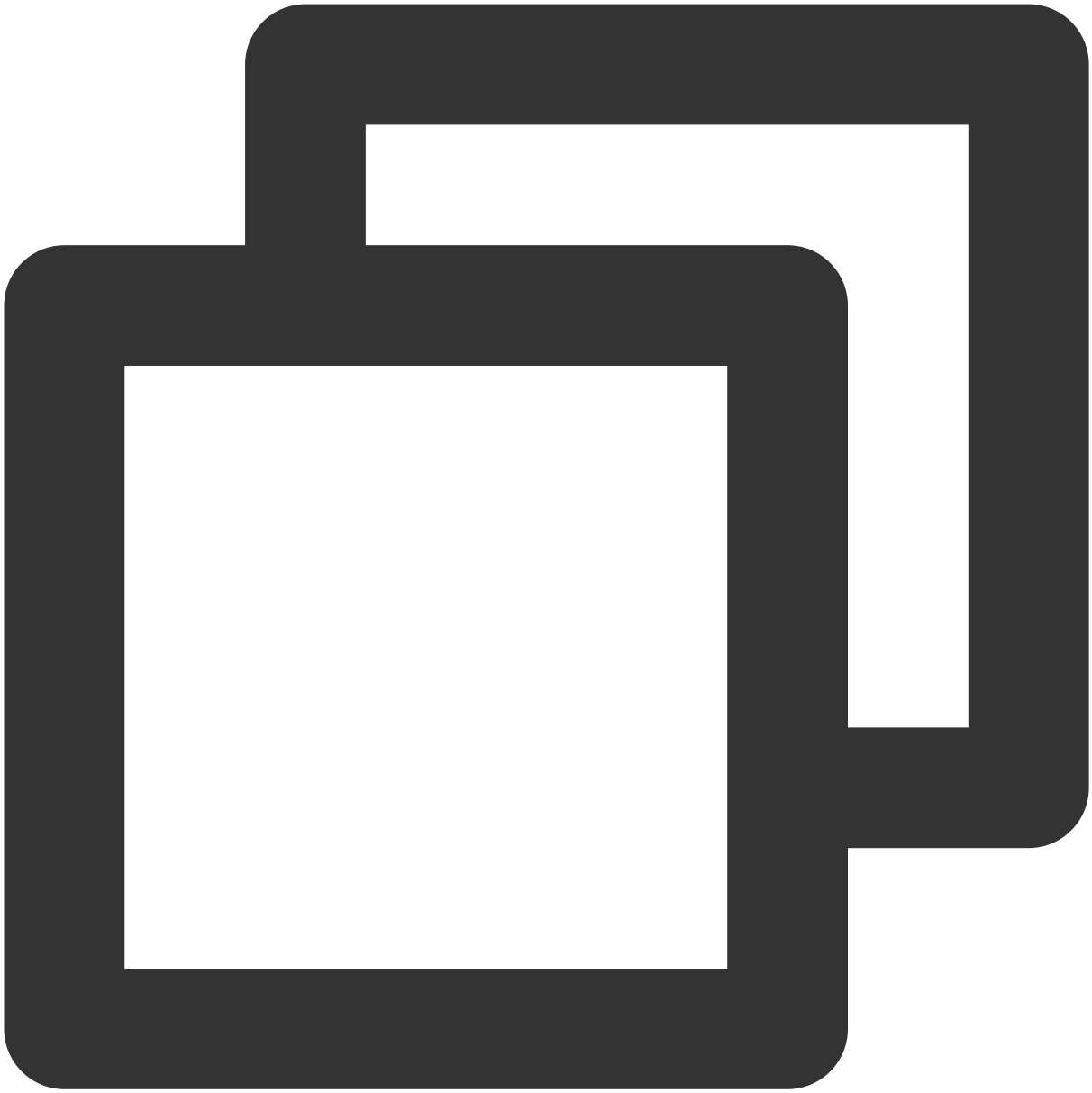
3. Drupal용 데이터베이스(이름: drupal)를 생성합니다.



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

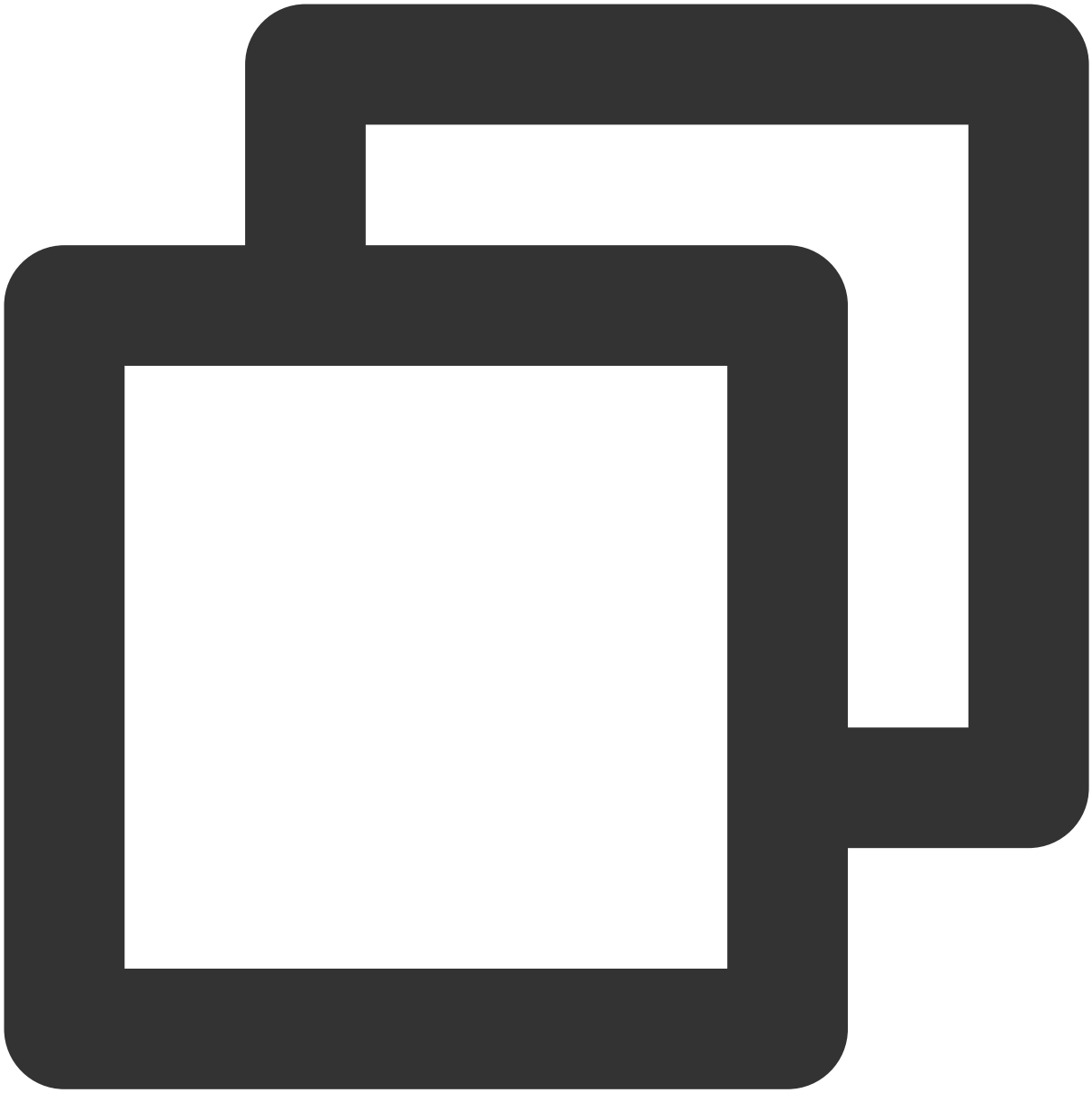
여기서 drupal은 Drupal 서비스에서 사용되는 데이터베이스 이름입니다.

4. 데이터베이스에 사용자를 생성합니다.



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

사용자에게 권한을 부여하고 권한 부여가 성공한 후 데이터베이스를 종료합니다.

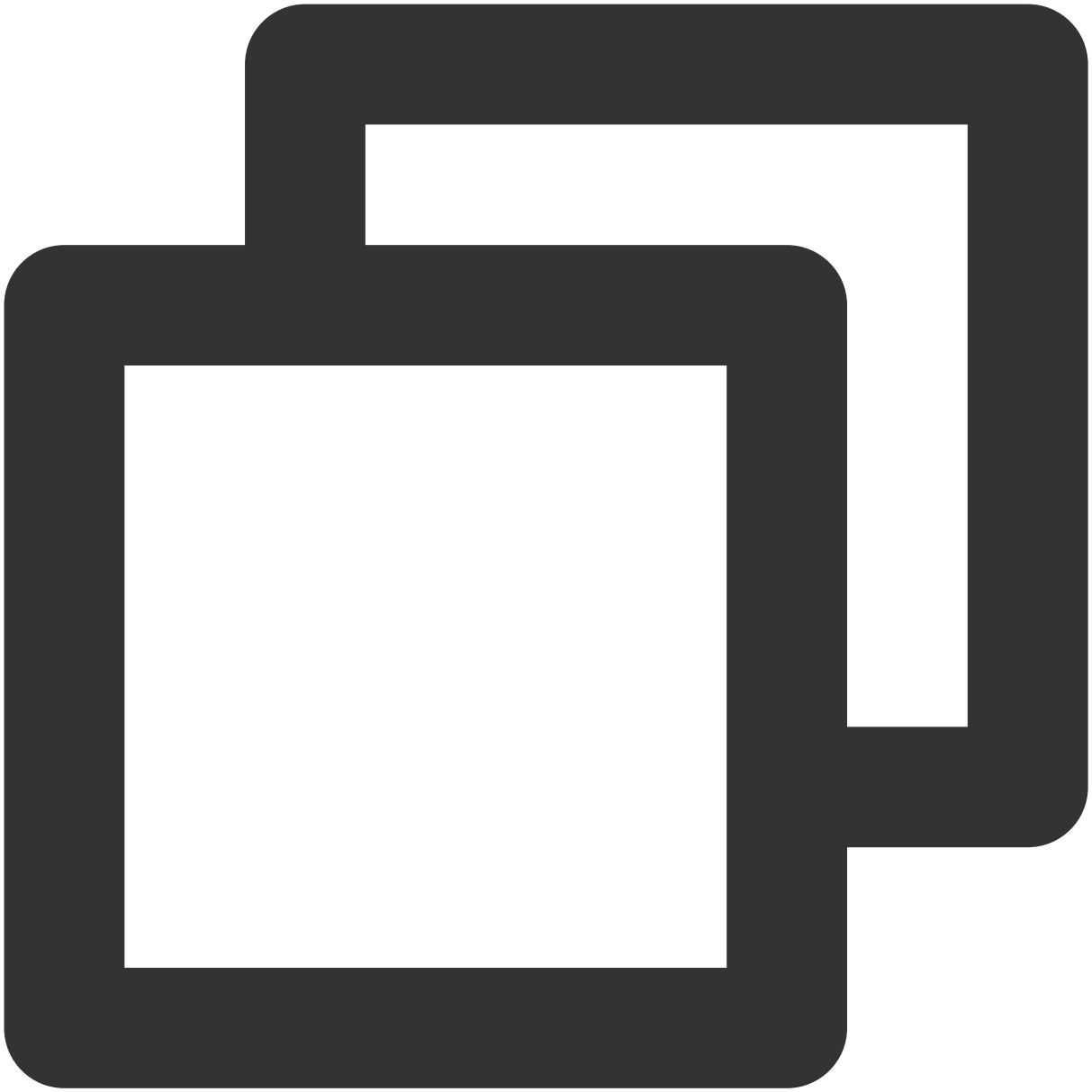


```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER, CREATE TEMPORARY
FLUSH PRIVILEGES;
exit
```

여기서 username은 Drupal 서비스에서 사용하는 데이터베이스 사용자 이름이며, password는 Drupal 서비스에서 사용하는 데이터베이스 비밀번호입니다.

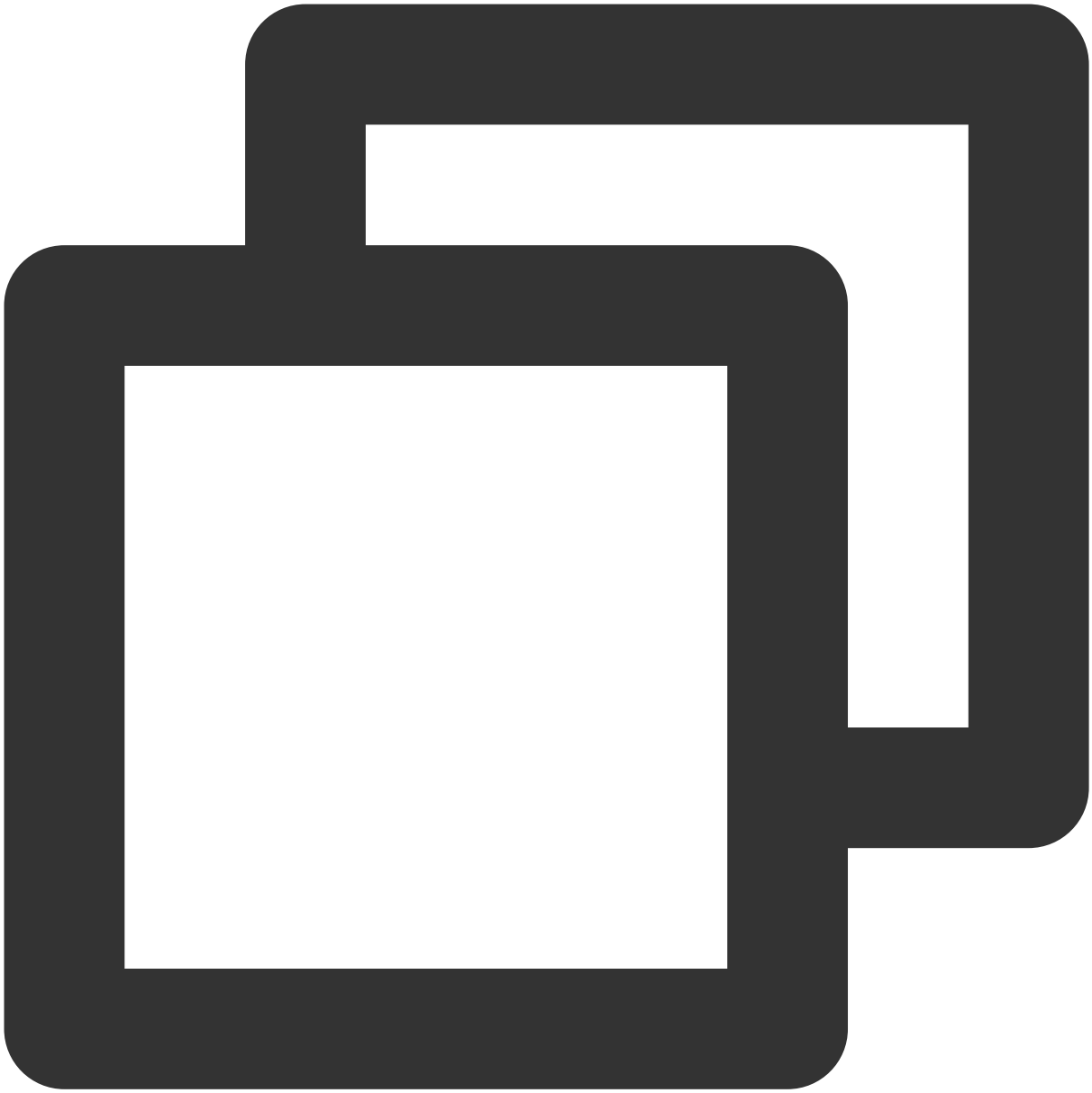
Apache 서비스 설치

1. CVM 인스턴스에 `yum` 을 사용해 Apache를 설치합니다.



```
yum install httpd -y
```

2. Apache 서비스를 실행합니다.



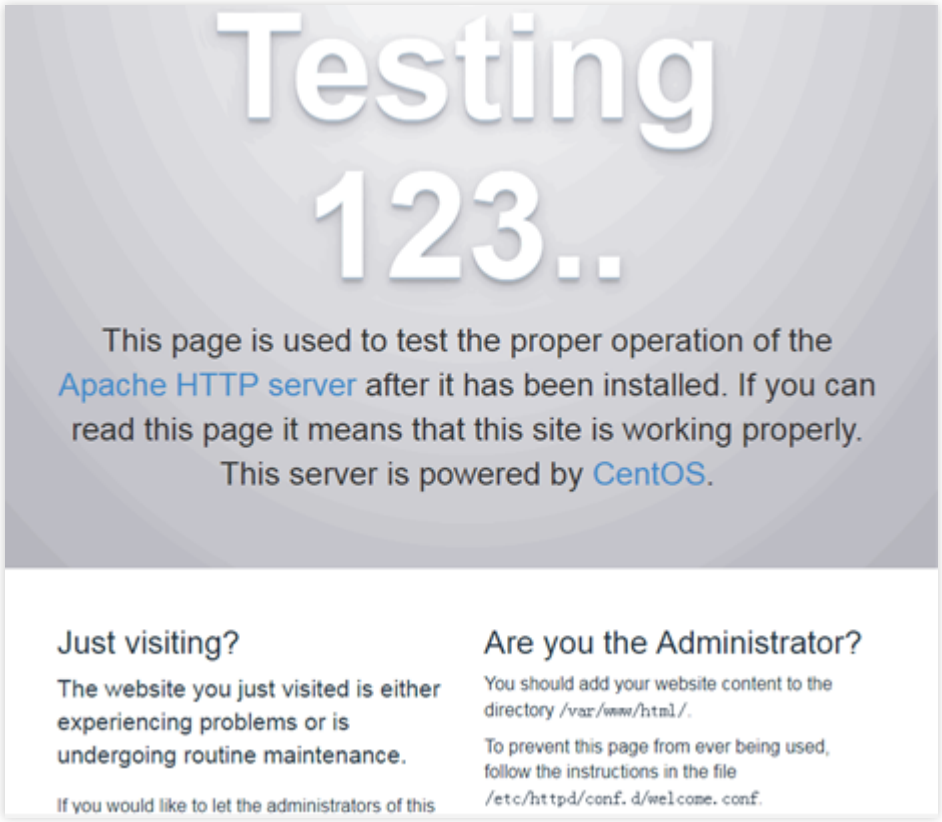
```
service httpd start
```

3. Apache를 테스트합니다.

주의 :

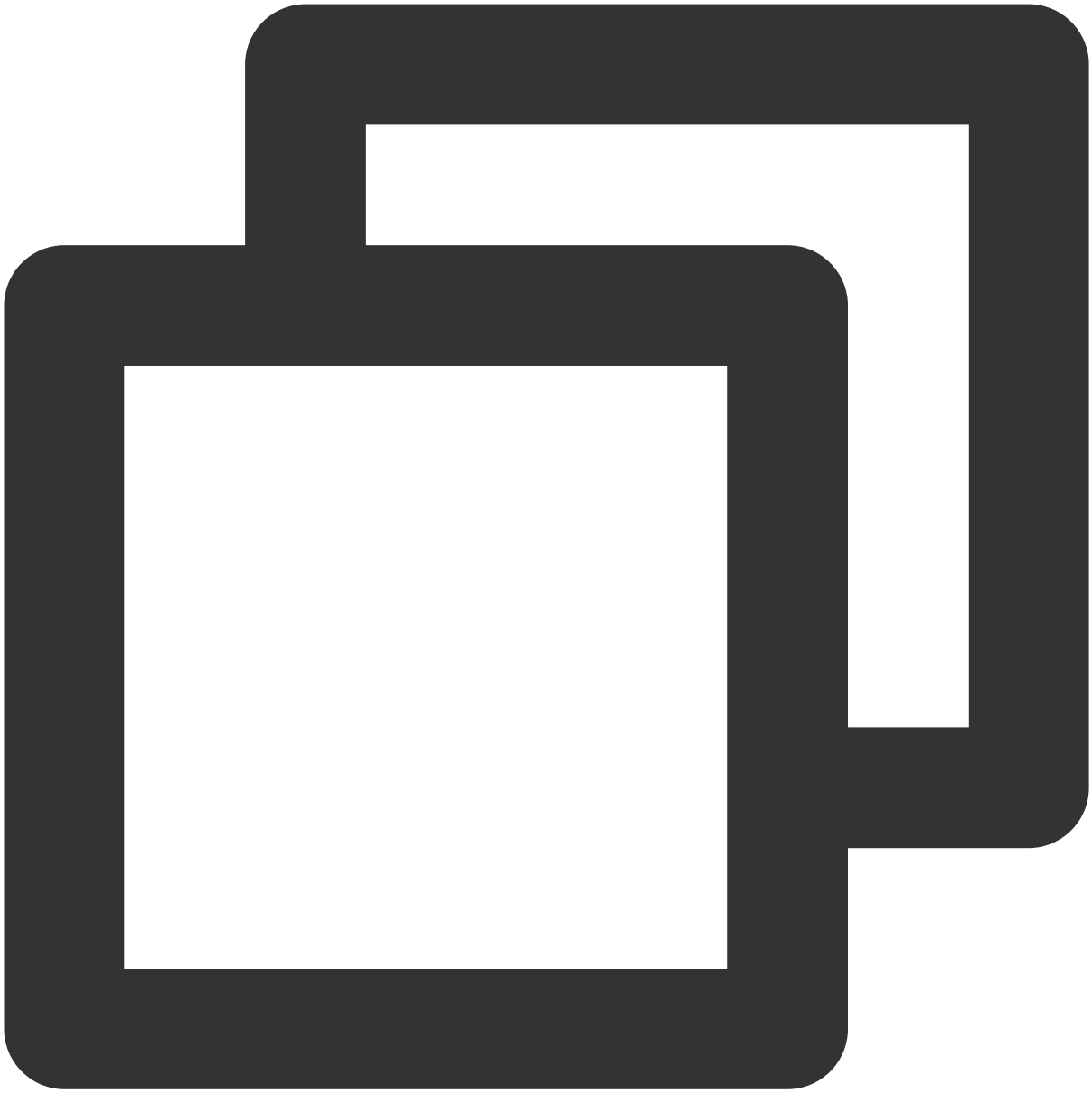
이 단계에서는 CVM 인스턴스의 보안 그룹에서 출처가 **all**이고 포트 프로토콜이 **TCP:80**인 인바운드 규칙을 구성해야 합니다. 보안 그룹을 구성하는 방법에 대한 자세한 내용은 보안 그룹을 참고하십시오.

로컬 브라우저에 `http://115.xxx.xxx.xxx/` 를 입력합니다(여기서 `115.xxx.xxx.xxx` 는 CVM 인스턴스의 공중망 IP). 다음 페이지가 나타나면 Apahce가 성공적으로 시작된 것입니다.



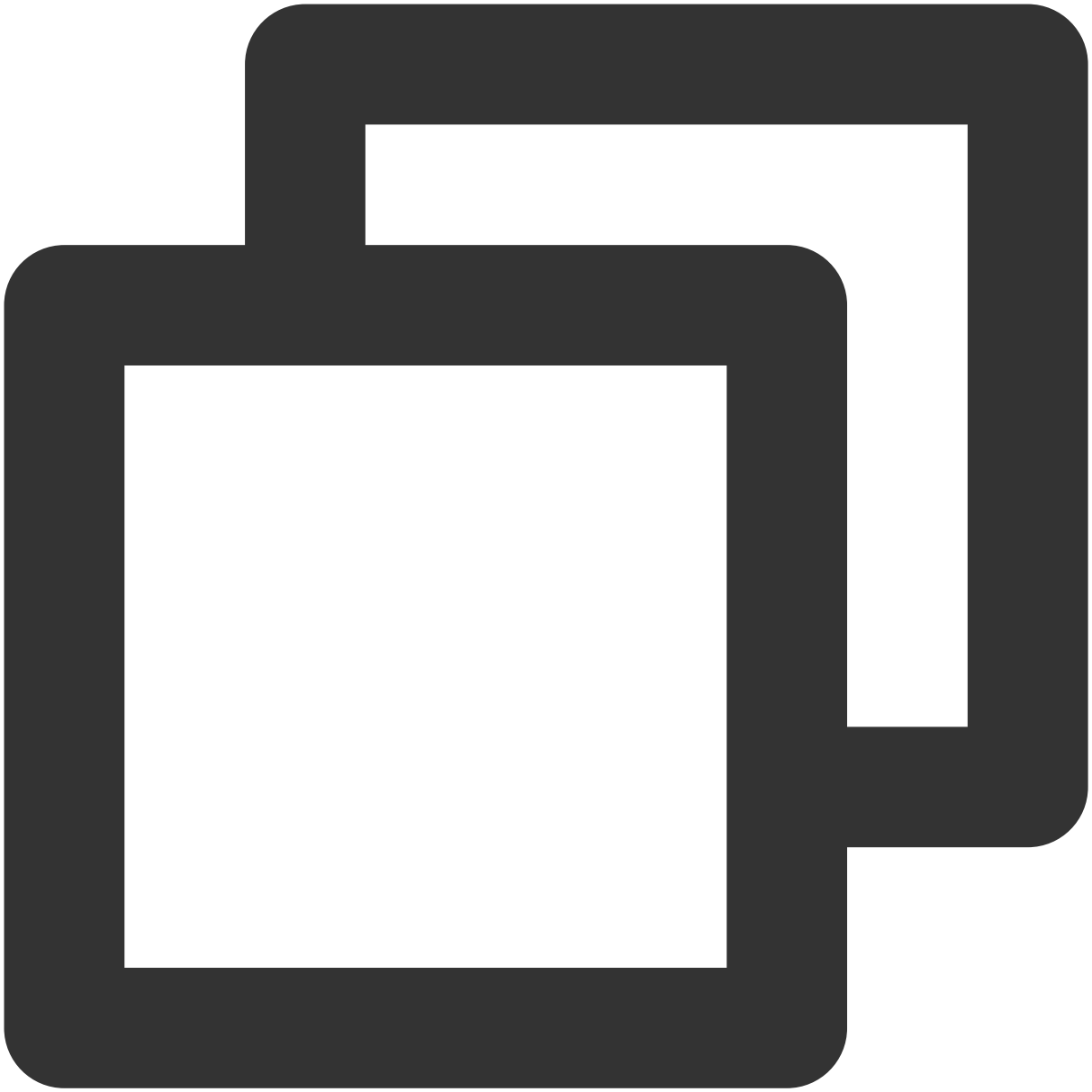
PHP 설치

1. `yum` 을 사용하여 CVM 인스턴스에 PHP 및 해당 확장을 설치합니다.



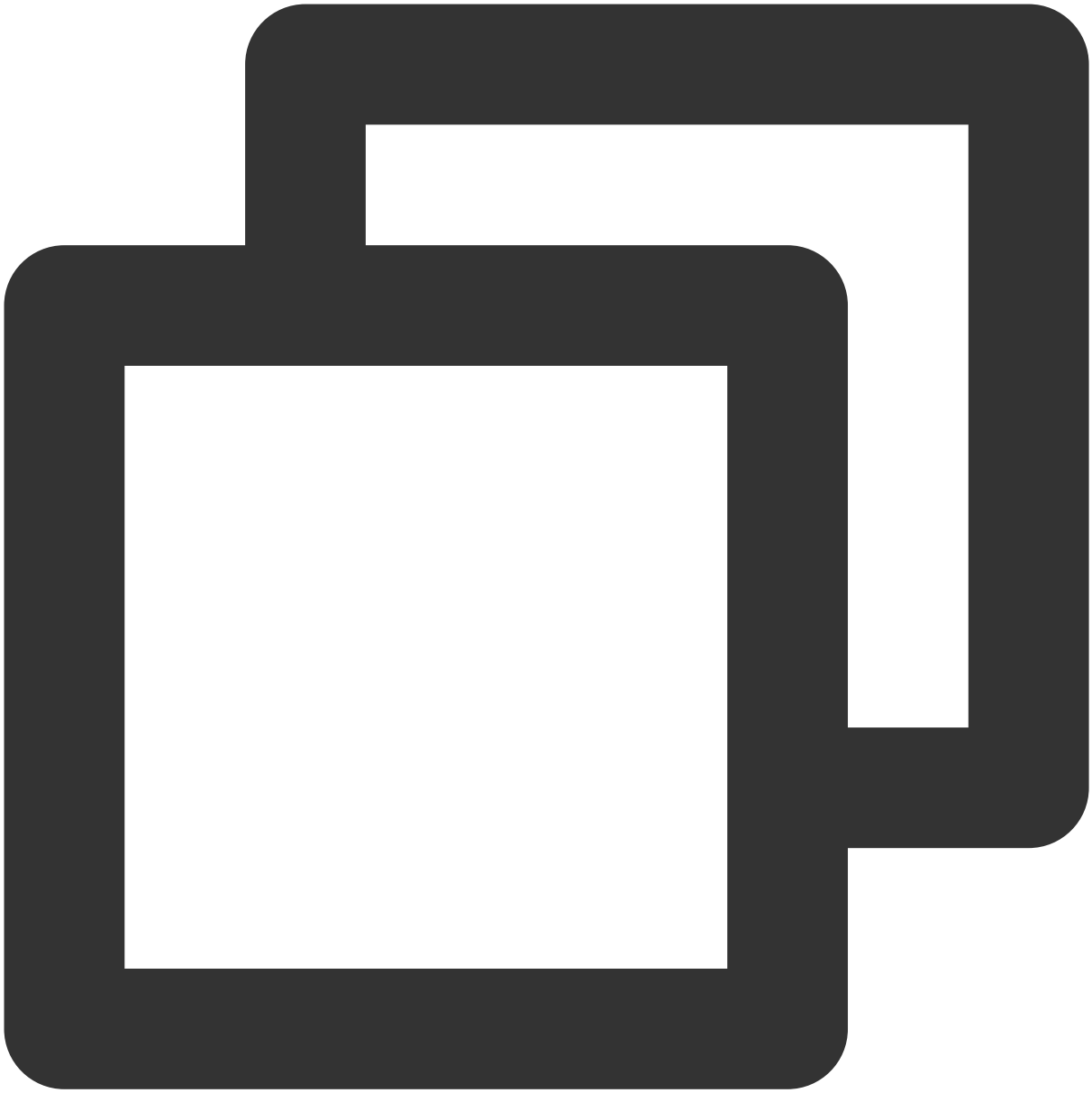
```
yum install php php-dom php-gd php-mysql php-pdo -y
```

2. 아래의 예시 코드와 같이, CVM의 `/var/www/html` 디렉터리에 info.php 파일을 생성하여 PHP가 성공적으로 설치되었는지 확인합니다.



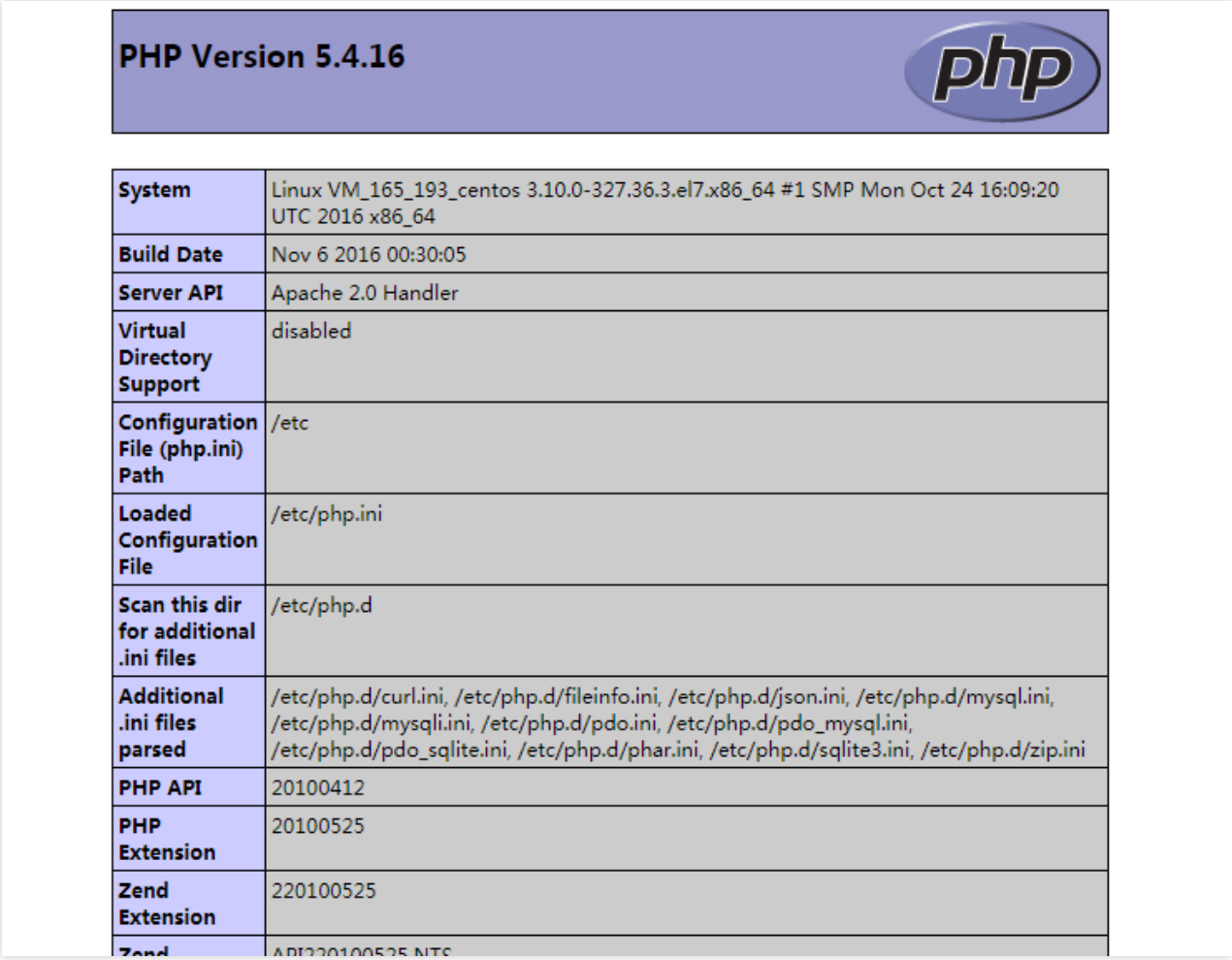
```
<?php phpinfo(); ?>
```

3. Apache 서비스를 재시작합니다.



```
service httpd restart
```

4. 로컬 브라우저에 `http://115.xxx.xxx.xxx/info.php` 를 입력합니다(여기서 `115.xxx.xxx.xxx` 는 CVM 인스턴스의 공중망 IP). 다음 페이지가 나타나면 PHP가 성공적으로 설치된 것입니다.

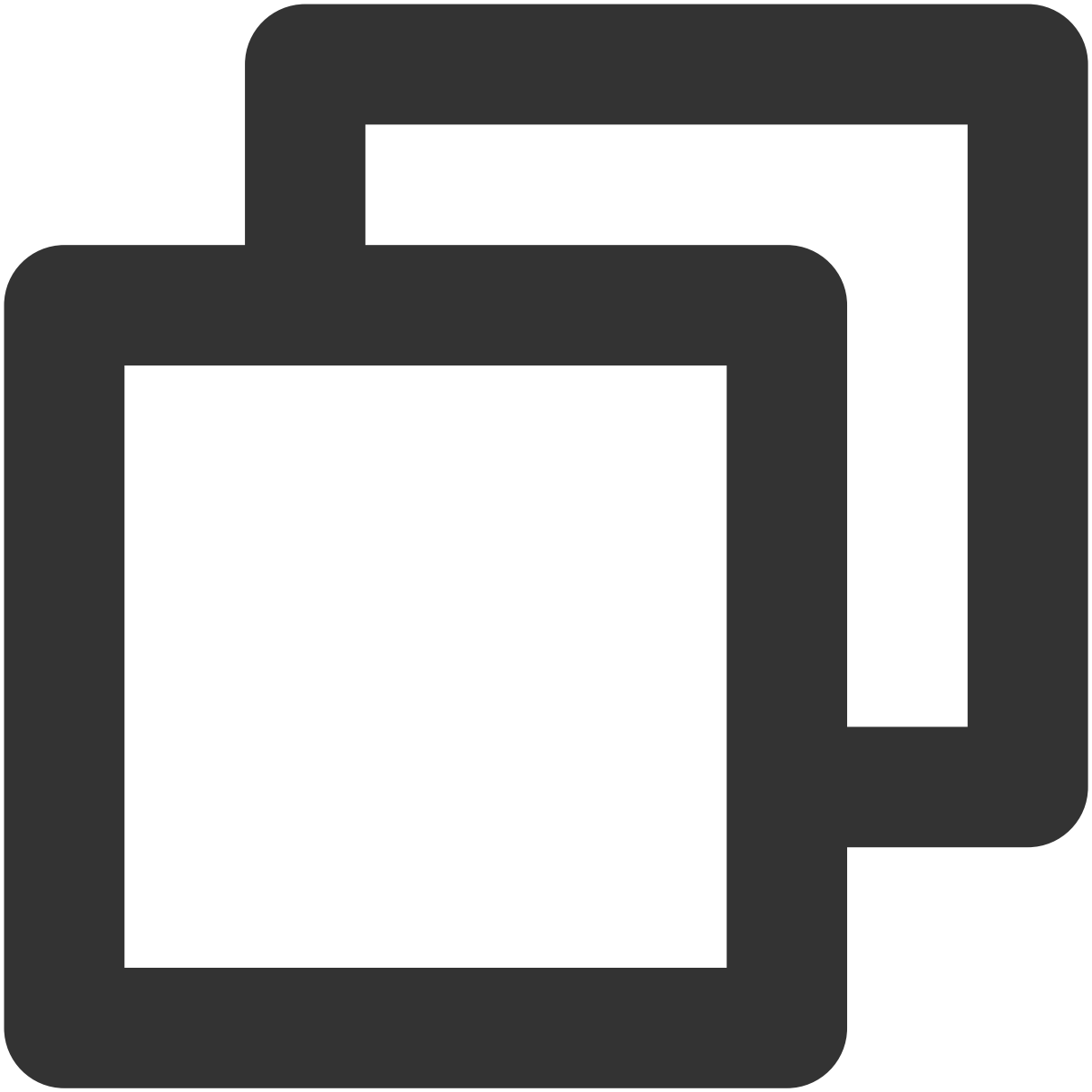


PHP Version 5.4.16

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

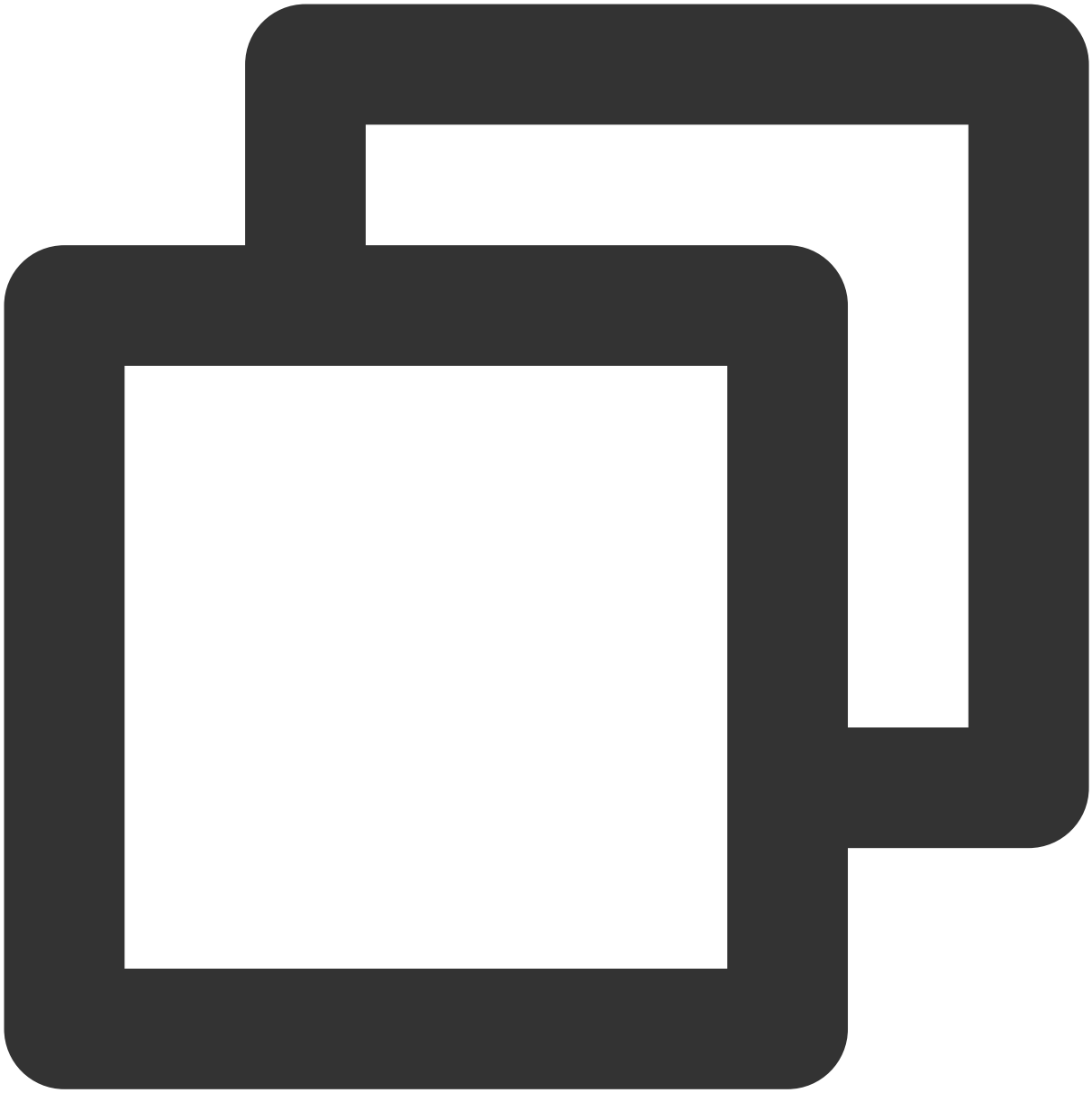
Drupal 서비스 설치

1. Drupal 설치 패키지를 다운로드합니다.



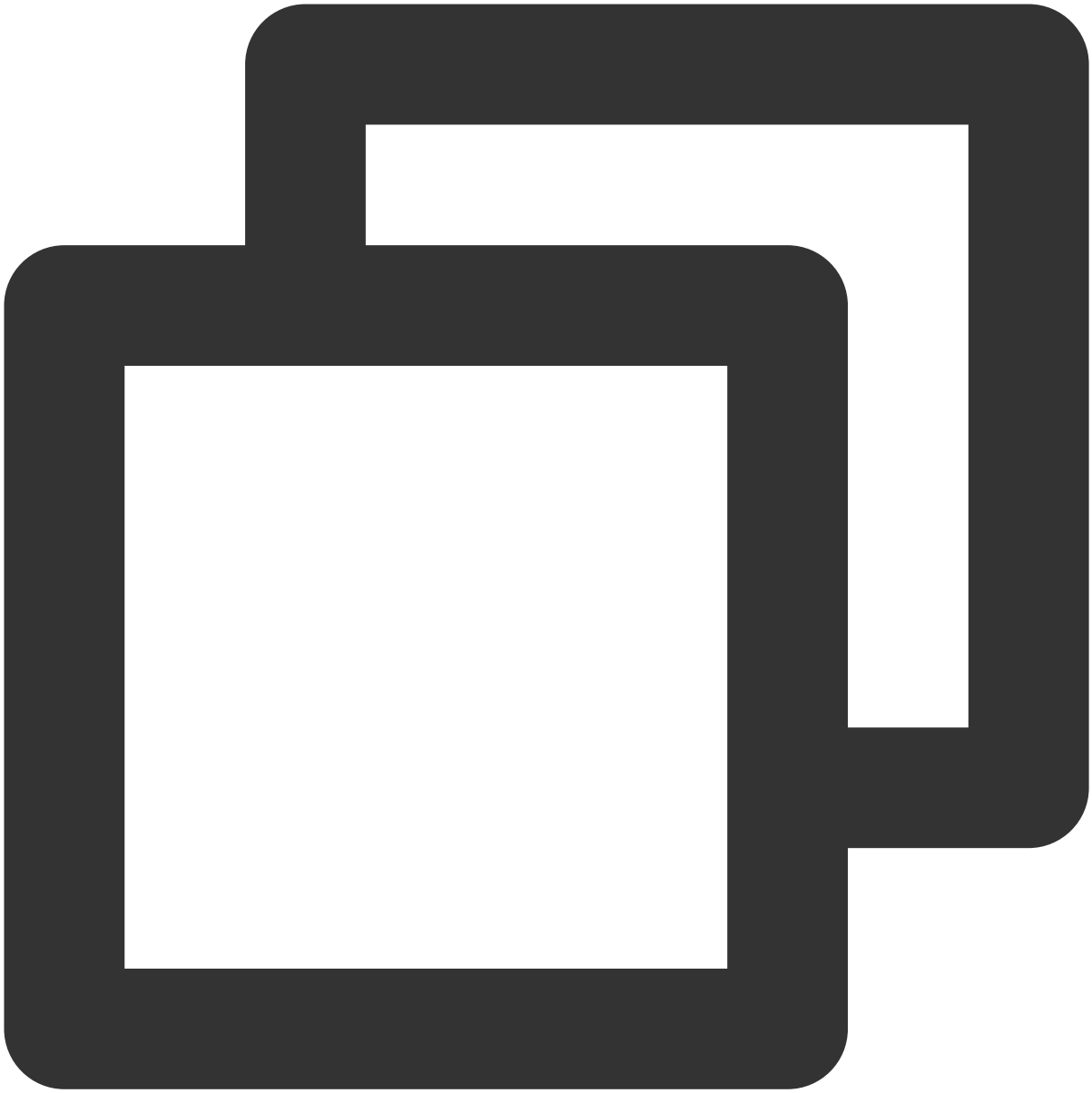
```
wget http://ftp.drupal.org/files/projects/drupal-7.56.zip
```

2. 웹 사이트의 루트 디렉터리에 압축을 해제합니다.



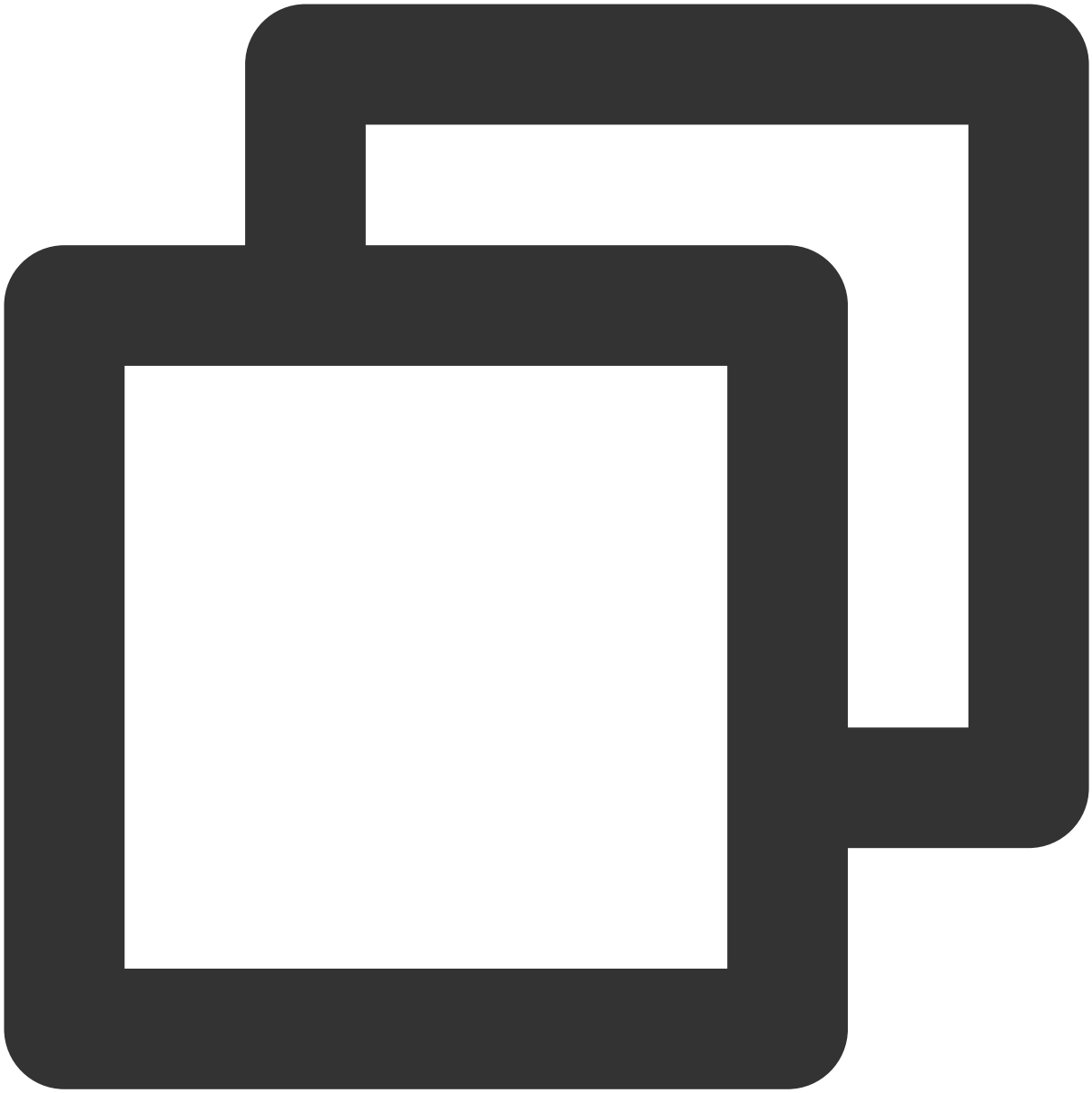
```
unzip drupal-7.56.zip
mv drupal-7.56/* /var/www/html/
```

3. 중국어 번역 키트를 다운로드합니다.



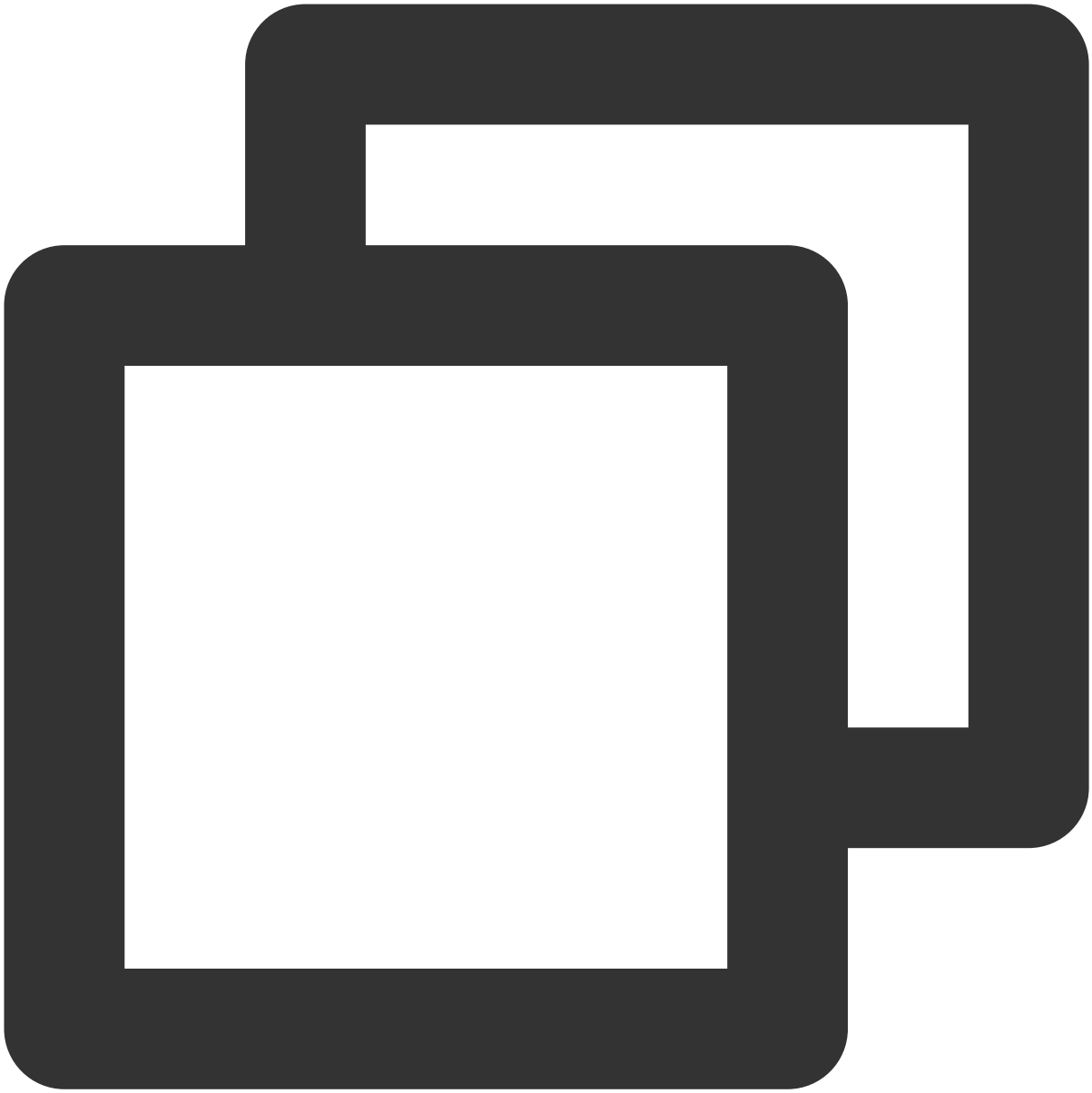
```
cd /var/www/html/
wget -P profiles/standard/translations http://ftp.drupal.org/files/translations/7.x
```

4. `sites` 디렉터리가 속한 소유자 및 그룹을 수정합니다.



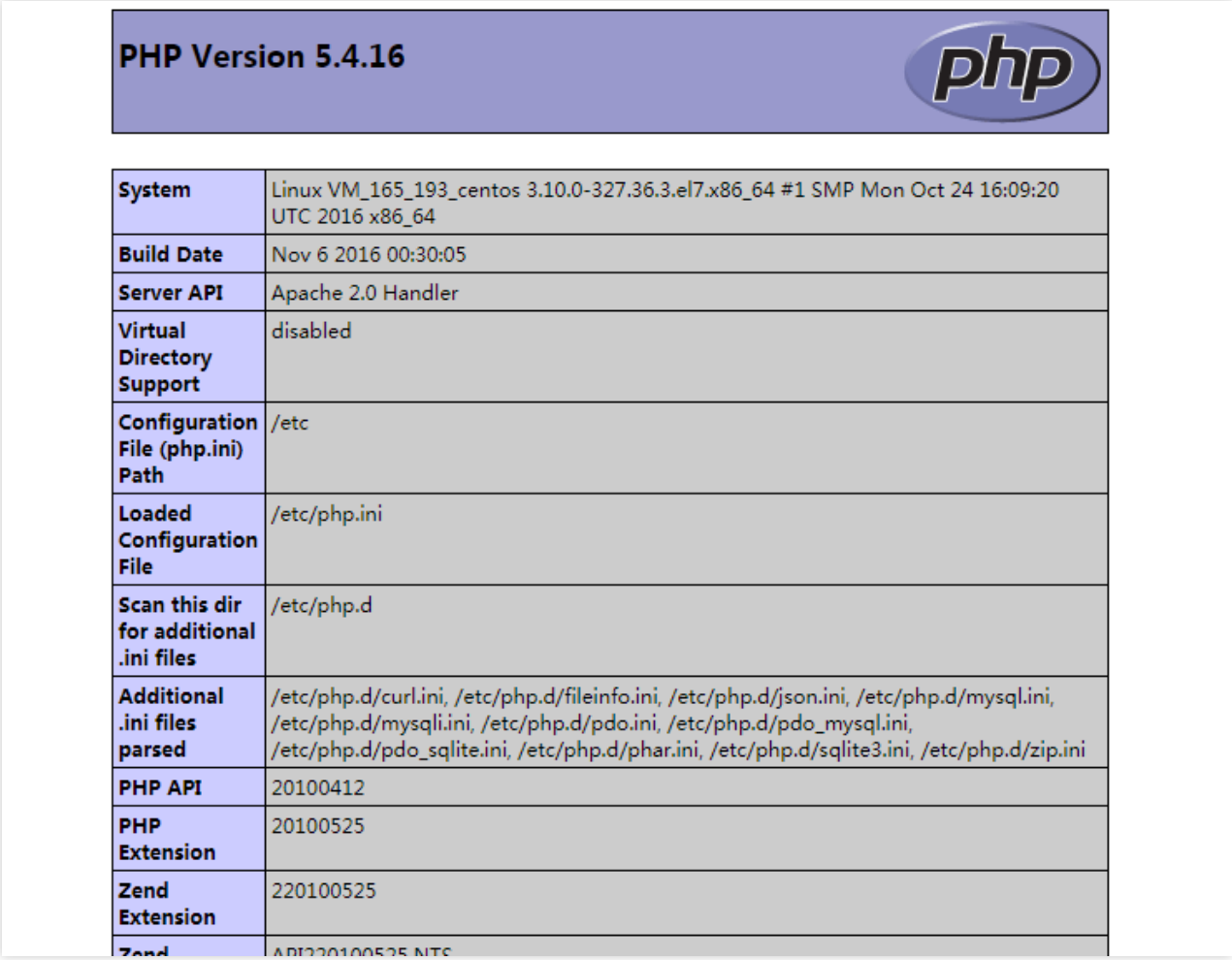
```
chown -R apache:apache /var/www/html/sites
```

5. Apache 서비스를 재시작합니다.



```
service httpd restart
```

6. 로컬 브라우저에 `http://115.xxx.xxx.xxx/` (여기서 `115.xxx.xxx.xxx` 는 CVM 인스턴스의 공중망 IP)를 입력하여 Drupal의 설치 인터페이스로 이동하고 설치할 버전을 선택한 다음 [Save and continue]를 클릭합니다.



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

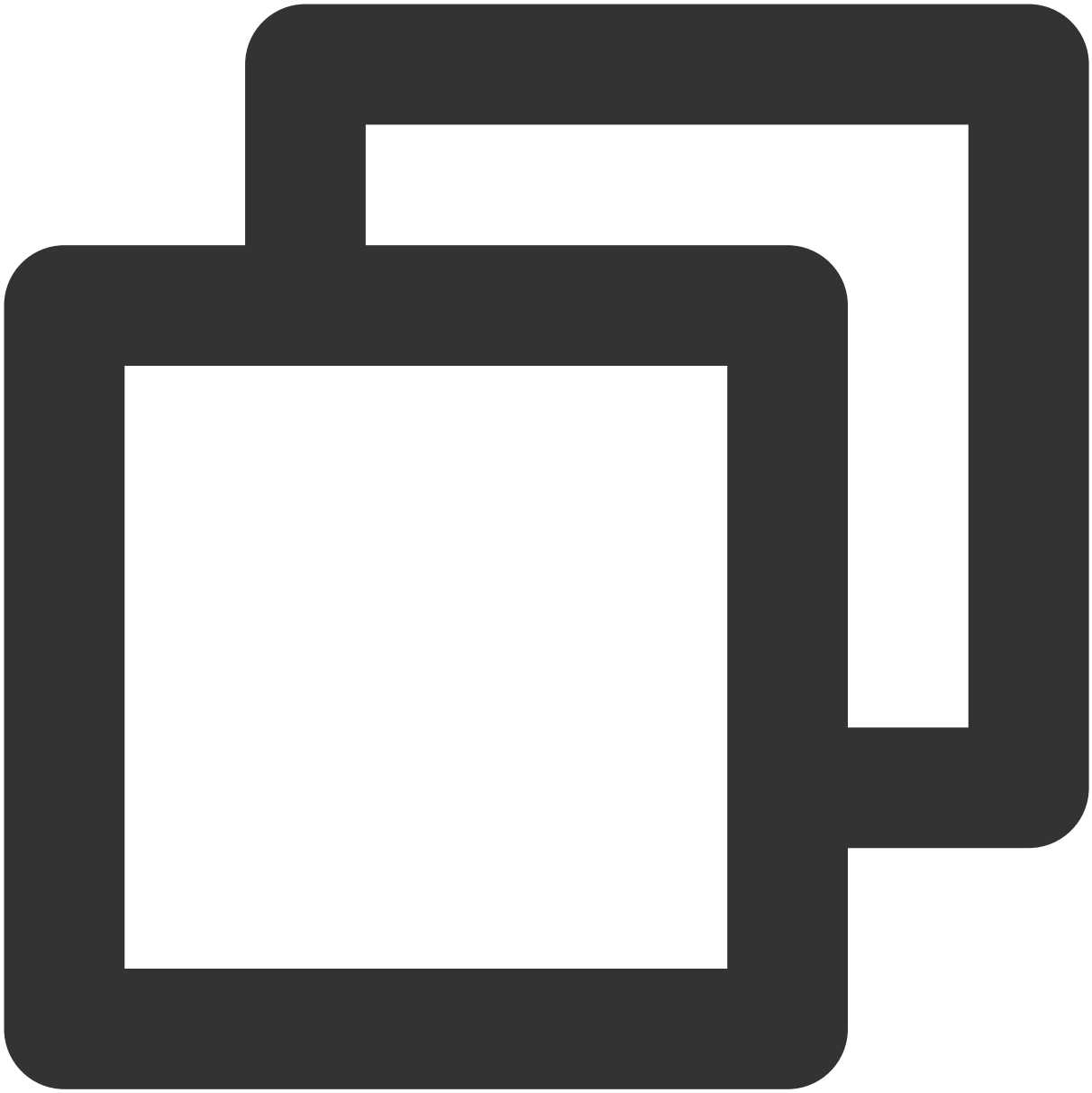
7. 설치 언어를 선택하고, [Save and continue]를 클릭합니다.
8. 데이터베이스를 설정하고, **mariadb 서비스 설치**에서 구성한 데이터베이스 정보를 입력합니다.
9. 사이트 정보를 입력합니다.
10. Drupal 설치를 완료합니다.
11. `http://115.xxx.xxx.xxx/` (여기서 `115.xxx.xxx.xxx` 는 CVM 인스턴스의 공중망 IP)에 액세스하여 웹 사이트를 사용자 지정할 수 있습니다.

Python을 통해 MySQL API 사용
인스턴스 구매

최종 업데이트 날짜: : 2024-07-25 16:38:48

API	설명
CreateDBInstance	CDB 인스턴스 생성
CreateDBInstanceHour	CDB 인스턴스 생성(종량제)
DescribeDBInstances	인스턴스 리스트 조회
DescribeDBPrice	데이터베이스 가격 조회
DescribeDBZoneConfig	CDB 판매 규격 획득
InitDBInstances	신규 인스턴스 초기화

CreateDBInstance CDB 인스턴스 생성



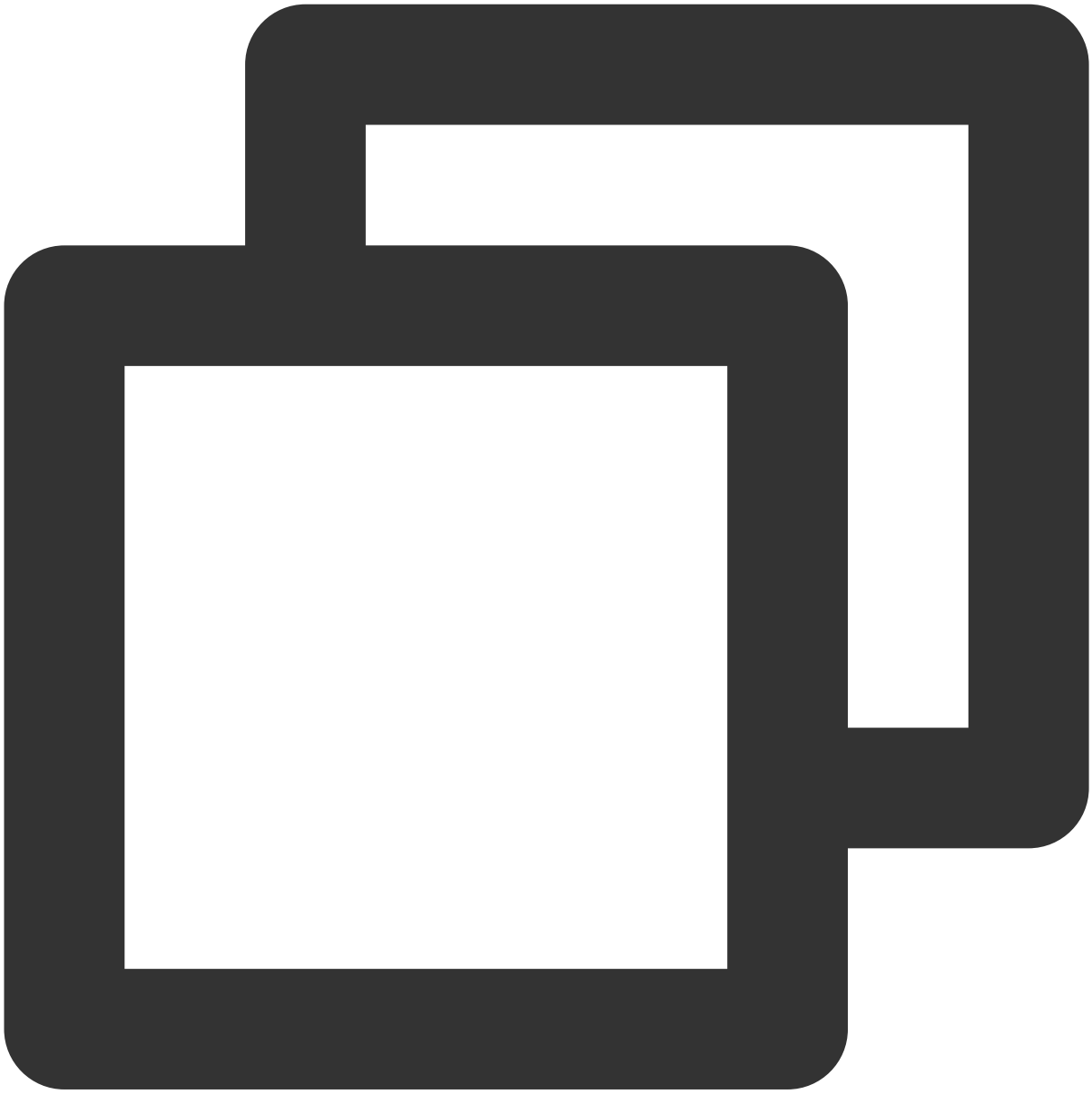
```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# 클라우드 API 게이트 모듈 불러오기
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
```

```
'''마스터 인스턴스 구매'''
def CreateDBInstancedemomaster():
    try:
        # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secr
        cred = credential.Credential("secretId", "secretKey")

        #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
        client = cdb_client.CdbClient(cred, "ap-beijing")

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


        # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
        resp = client.CreateDBInstance(req)
        # json 포맷의 문자열 출력
        print(resp.to_json_string())


    except TencentCloudSDKException as err:
        msg = traceback.format_exc() # 방식1
        print (msg)




'''읽기 전용 인스턴스 구매'''
def CreateDBInstancedemoro():
    try:
        # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secr
        cred = credential.Credential("secretId", "secretId")

        #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
        client = cdb_client.CdbClient(cred, "ap-beijing")
```

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


        # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
        resp = client.CreateDBInstance(req)
        # json 포맷의 문자열 출력
        print(resp.to_json_string())


    except TencentCloudSDKException as err:
        msg = traceback.format_exc() # 방식1
        print (msg)


'''재해 복구 인스턴스 구매'''
def CreateDBInstancedemodr():
    try:
        # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secr
        cred = credential.Credential("secretId", "secretKey")

        #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
        client = cdb_client.CdbClient(cred, "ap-shanghai")

        #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
        req = models.CreateDBInstanceRequest()
```

```
        req.Memory = 4000
        req.Volume = 200
        req.Period = 1
        req.GoodsNum =1
        #req.Zone = "ap-shanghai-2"
        req.Port = 3306
        req.InstanceRole = "dr"
        #req.MasterInstanceId
        req.EngineVersion = "5.6"
        req.Password = "CDB@Qcloud"
        req.ProtectMode = 0
        req.DeployMode =0
        #req.SlaveZone = "ap-guangzhou-3"
        req.ParamList = [{"name":"max_connections","value":"1000"},{"name":"lower_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 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
        resp = client.CreateDBInstance(req)
        # json 포맷의 문자열 출력
        print(resp.to_json_string())


    except TencentCloudSDKException as err:
        msg = traceback.format_exc() # 방식1
        print (msg)

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

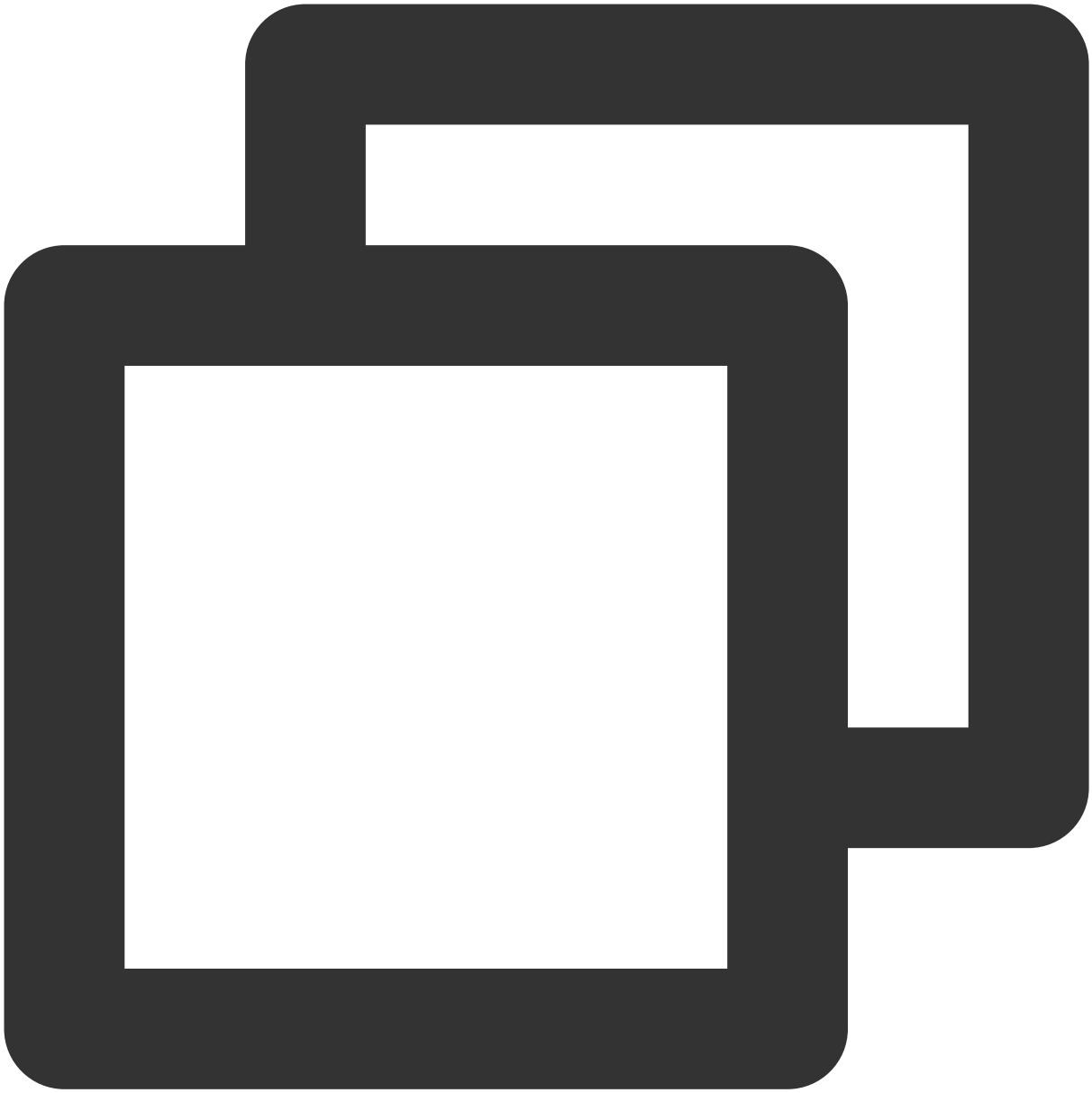
CreateDBInstanceHour CDB 인스턴스 생성(종량제)



```
'''시간당 과금의 경우 동결 예치금이 필요하며 계정에 금액이 예치되어 있어야 함. 계정 잔액이 0인 
#!/usr/bin/python
# -*- coding: utf-8 -*-

# 클라우드 API 게이트 모듈 불러오기
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:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
    cred = credential.Credential("secretId", "secretKey")

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-beijing")

    #요청 객체 인스턴스화: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 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.CreateDBInstanceHour(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    msg = traceback.format_exc() # 방식1
    print (msg)
```

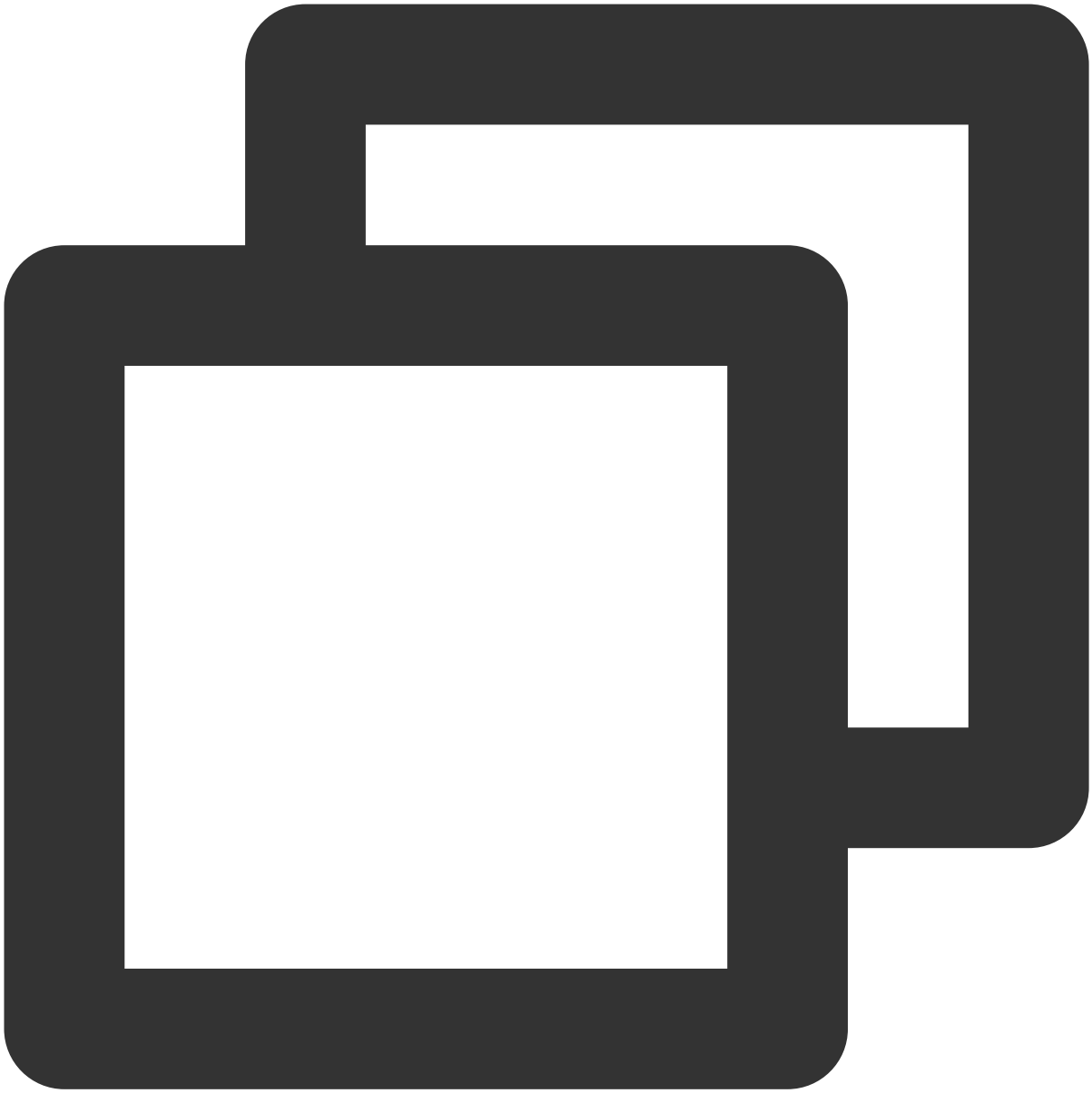
DescribeDBInstances 인스턴스 리스트 조회



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

# 클라우드 API 게이트 모듈 불러오기
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:
```

```
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
    cred = credential.Credential("secretId", "secretKey")

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBInstancesRequest()
    req.EngineVersions = ["5.6"]
    req.OrderBy = "instanceId"
    req.InstanceIds = ["cdb-1j8lumf6"]


    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.DescribeDBInstances(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
        msg = traceback.format_exc() # 방식1
        print (msg)
```

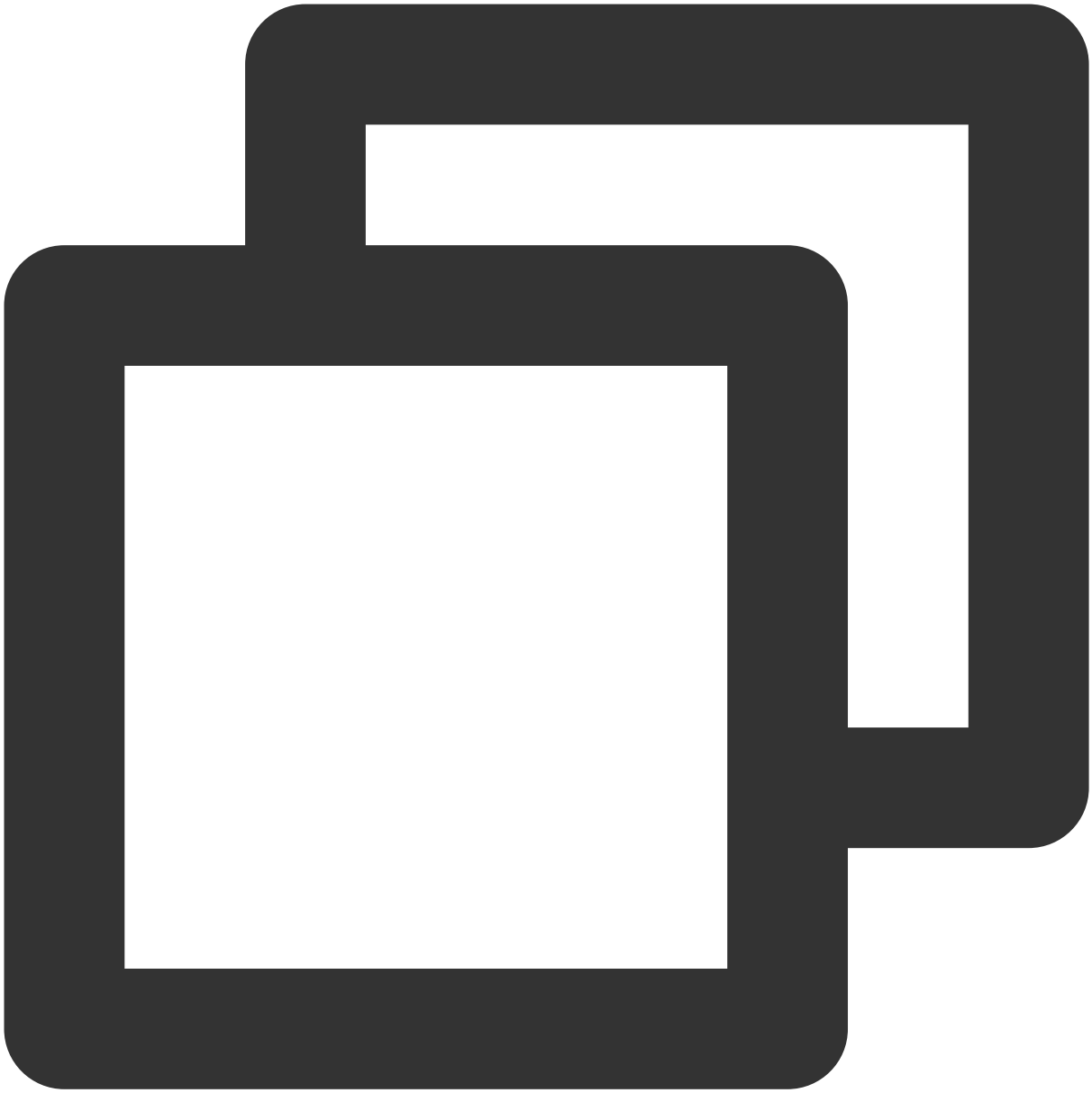
DescribeDBPrice 데이터베이스 가격 조회



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

# 클라우드 API 게이트 모듈 불러오기
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudS
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
    cred = credential.Credential("secretId", "secretKey")
```

```
    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-guangzhou")

    #요청 객체 인스턴스화: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 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.DescribeDBPrice(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

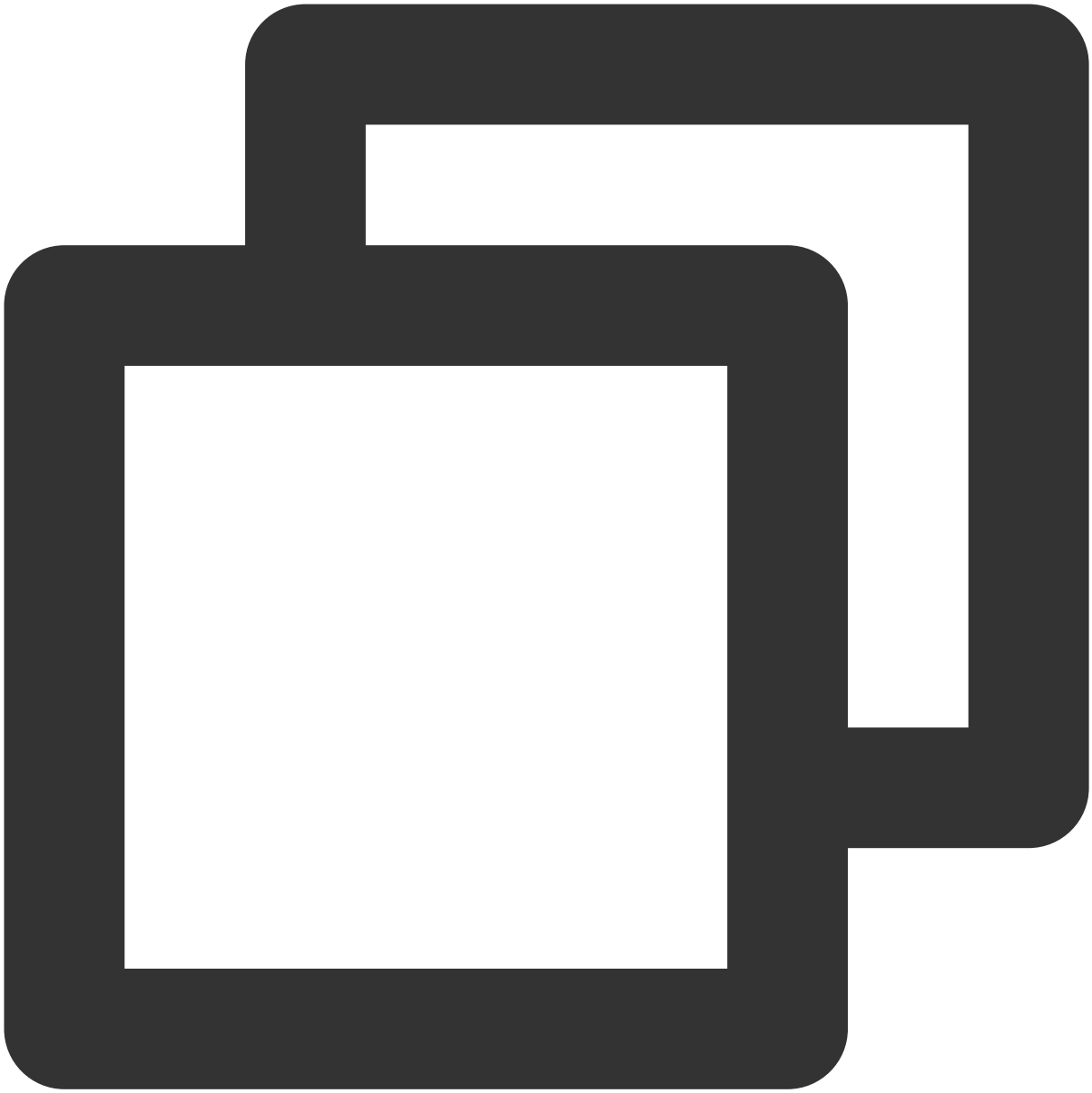
DescribeDBZoneConfig CDB 판매 규격 획득



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBZoneConfigRequest()
    #req.InstanceId = "cdb-j0edpju5"


    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.DescribeDBZoneConfig(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

InitDBInstances 신규 인스턴스 초기화



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
```

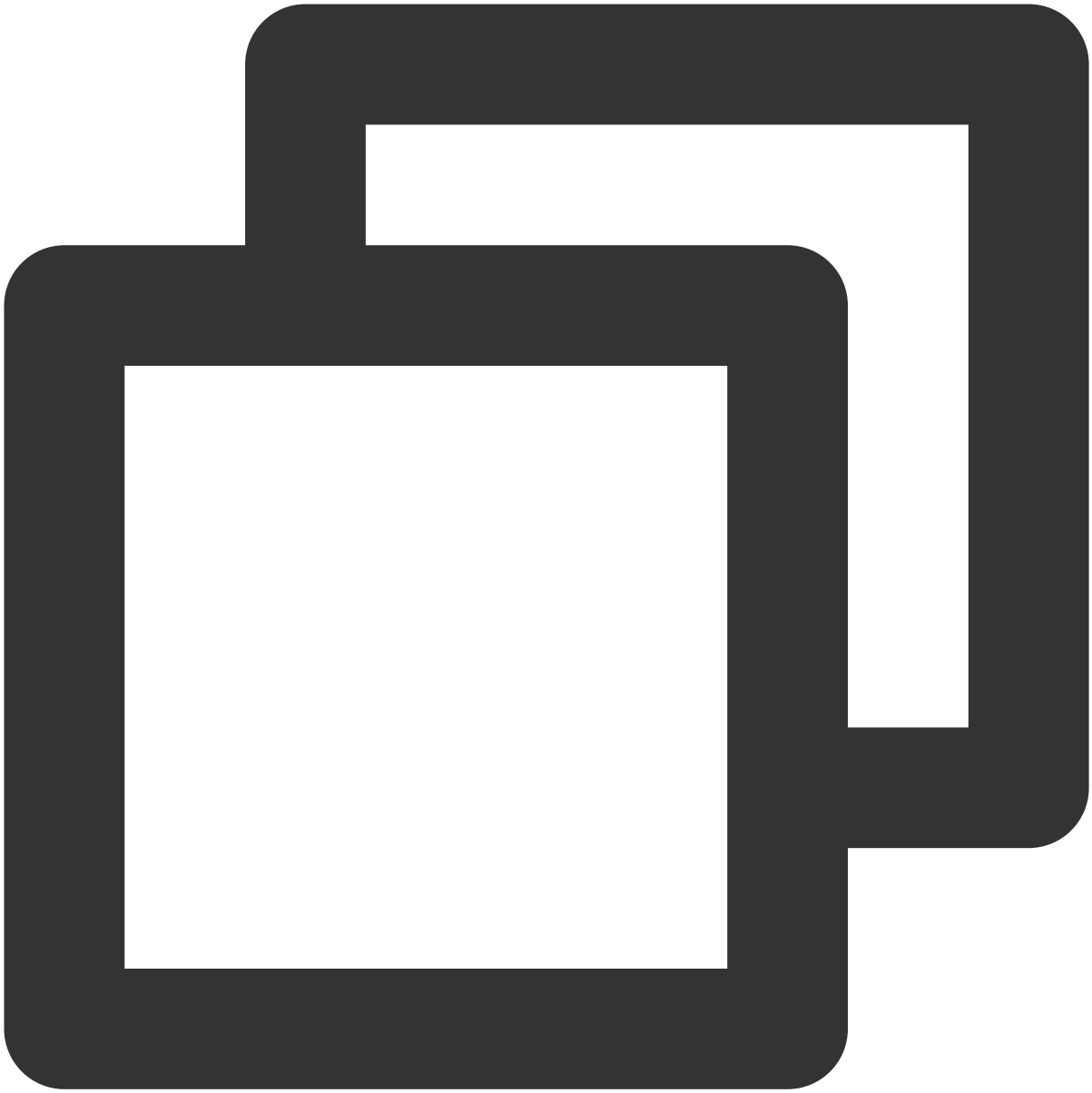
```
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
    cred = credential.Credential("secretId", "secretKey")

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화: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 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.InitDBInstances(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

인스턴스 관리

최종 업데이트 날짜: : 2024-07-25 16:38:48

API	설명
ModifyInstanceParam	인스턴스 매개변수 수정
CloseWanService	인스턴스 외부 네트워크 액세스 종료
OpenWanService	인스턴스 외부 네트워크 액세스 활성화
RestartDBInstances	인스턴스 재시작
OpenDBInstanceGTID	인스턴스 GTID 활성화
ModifyDBInstanceName	CDB 인스턴스 이름 수정
ModifyDBInstanceProject	CDB 인스턴스 서브 프로젝트 수정
ModifyDBInstanceVipVport	CDB 인스턴스 IP와 포트 번호 수정
DescribeDBInstanceCharset	CDB 인스턴스 문자 세트 조회
DescribeDBInstanceConfig	CDB 인스턴스 설정 정보 조회
DescribeDBInstanceGTID	CDB 인스턴스 GTID 활성화 여부 조회
DescribeDBInstanceRebootTime	CDB 인스턴스 예상 재시작 시간 조회

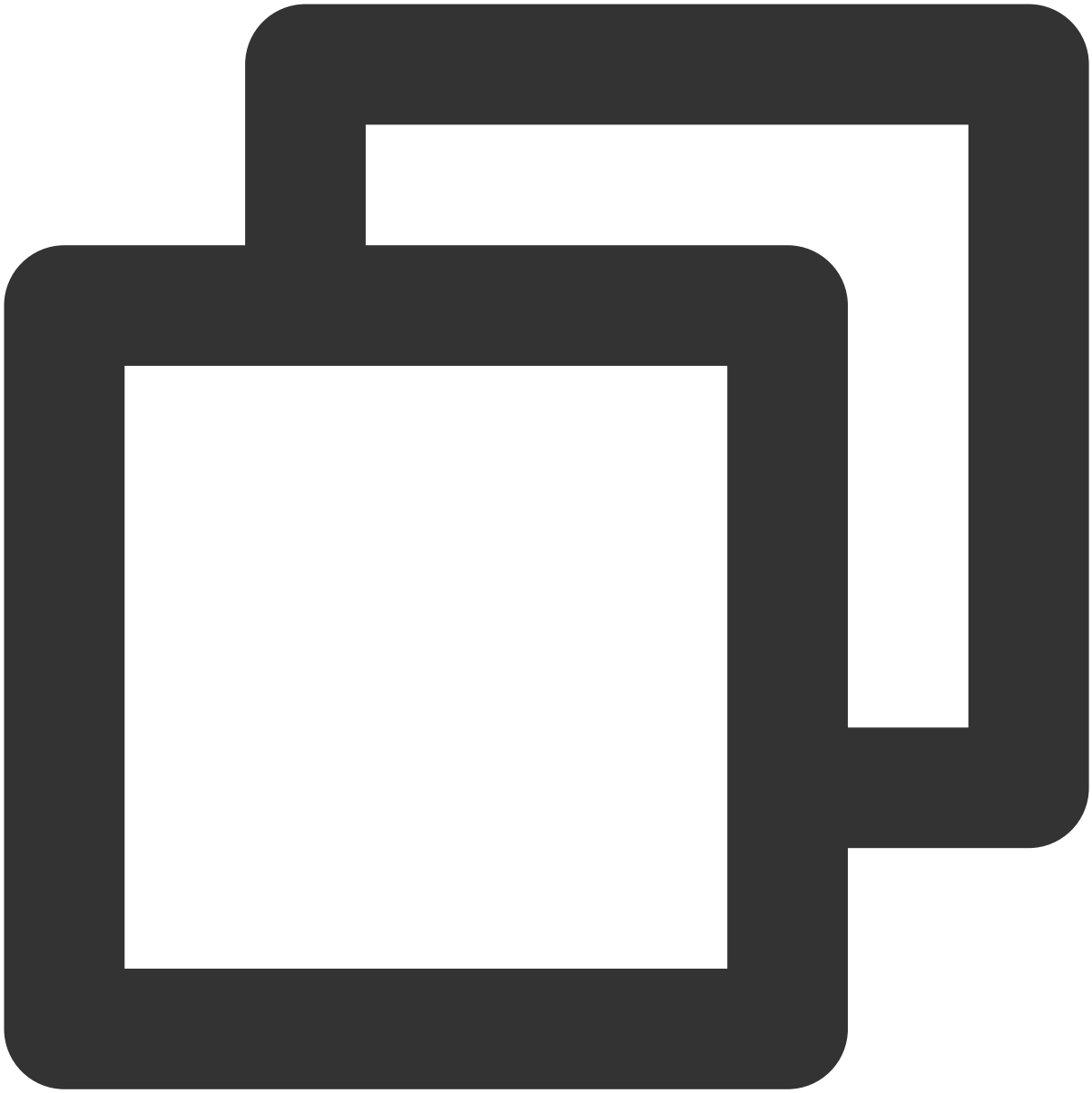
ModifyInstanceParam 인스턴스 매개변수 수정



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

# 클라우드 API 게이트 모듈 불러오기
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:
```

```
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
    cred = credential.Credential("secretId", "secretKey")

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화
    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 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.ModifyInstanceParam(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    msg = traceback.format_exc() # 방식1
    print (msg)
```

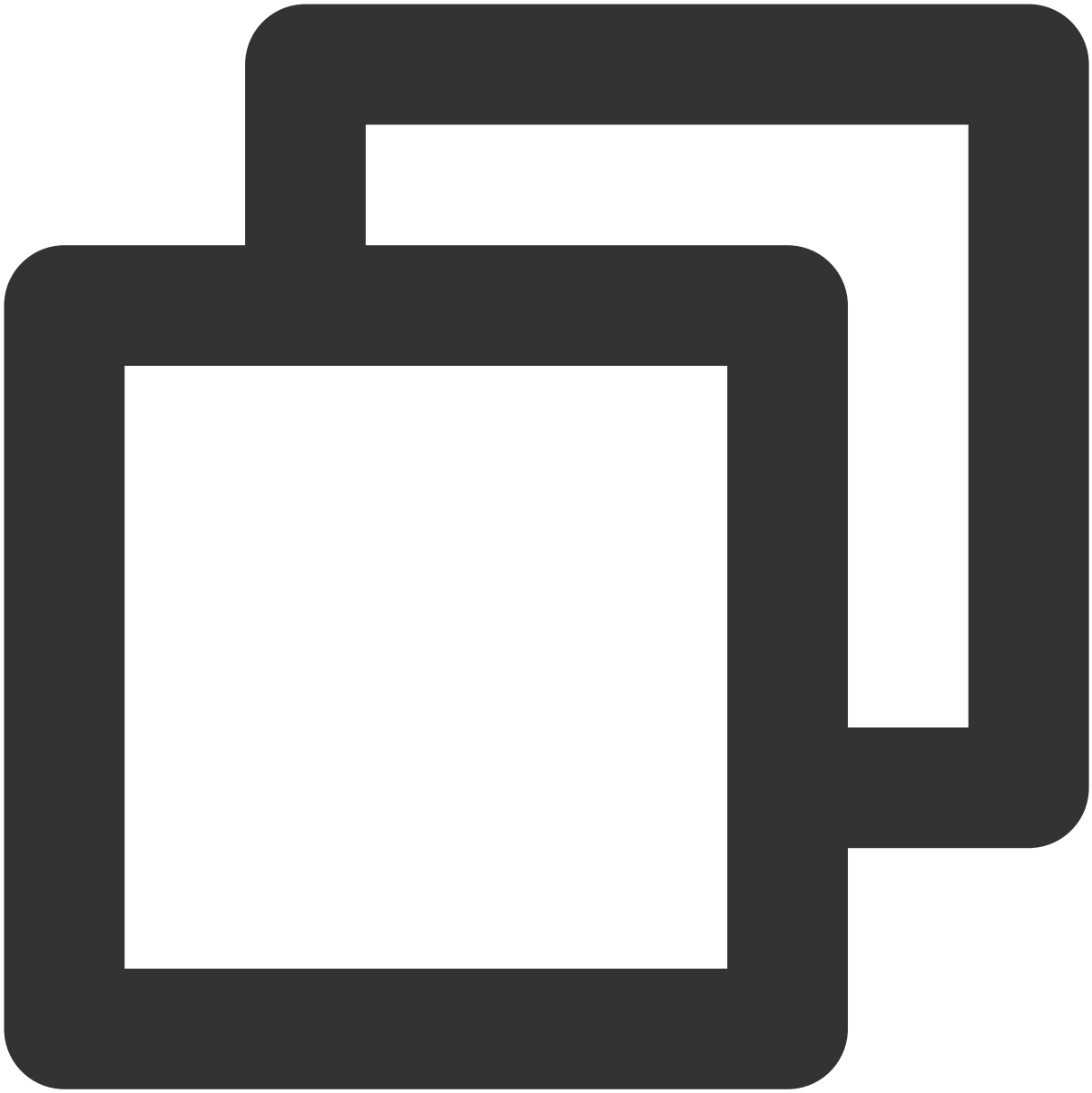
CloseWanService 인스턴스 외부 네트워크 액세스 종료



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.CloseWanServiceRequest()
    req.InstanceId = "cdb-1y6g3zj8"

    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.CloseWanService(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

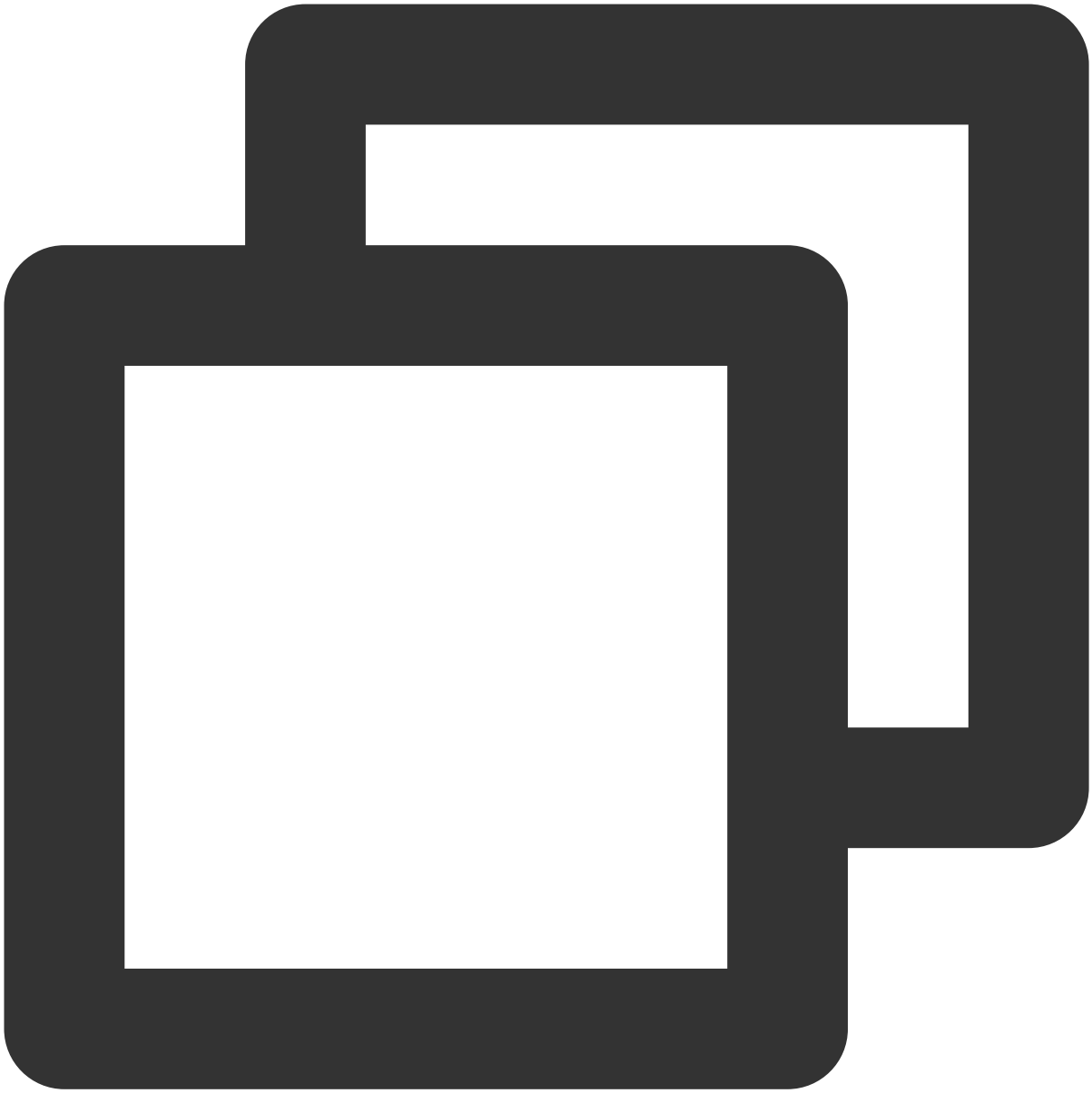
OpenWanService 인스턴스 외부 네트워크 액세스 활성화



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.OpenWanServiceRequest()
    req.InstanceId = "cdb-1y6g3zj8"

    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.OpenWanService(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

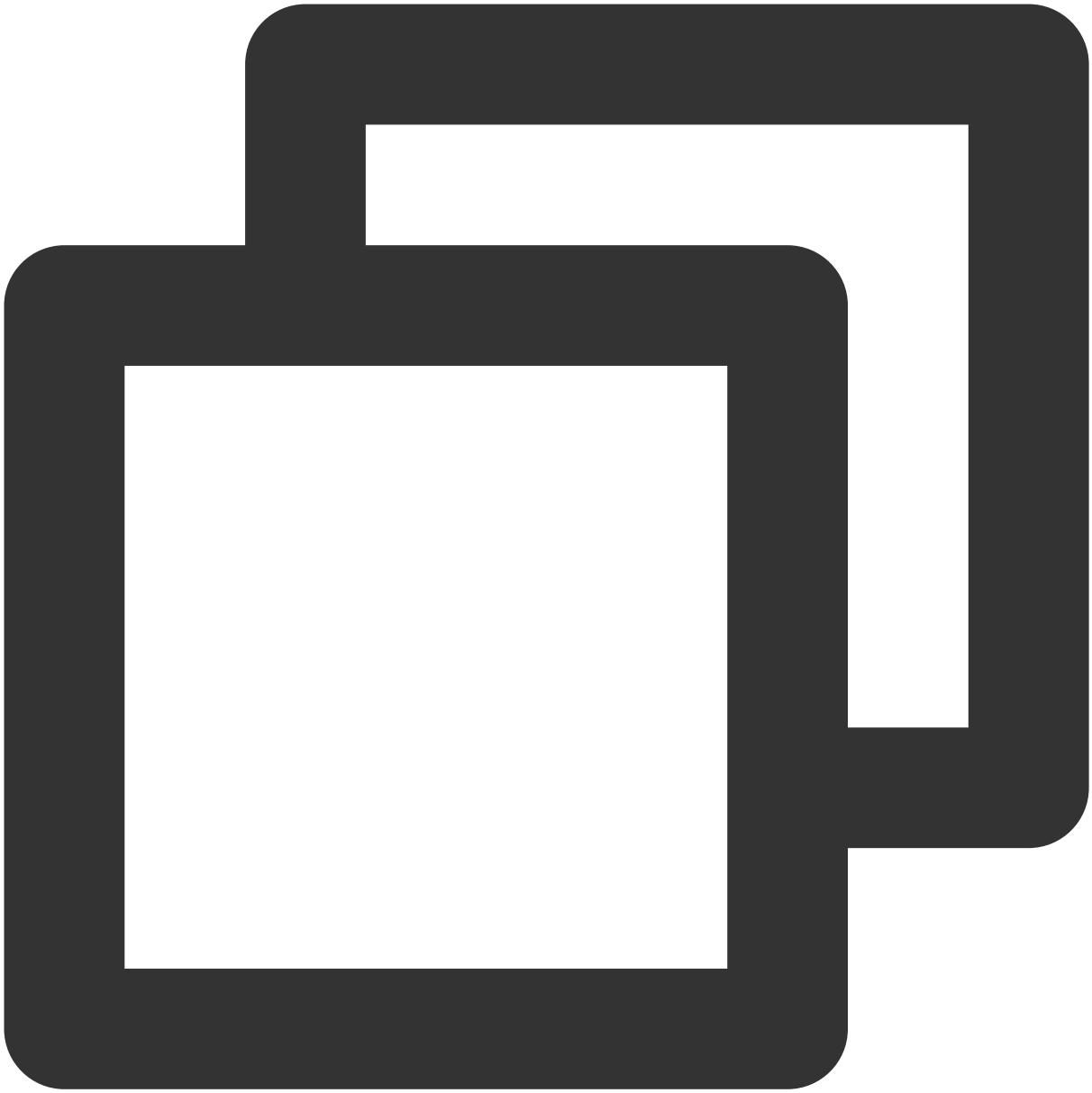
RestartDBInstances 인스턴스 재시작



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBInstancesRequest()
    #req.MasterInstanceId = "cdb-7ghaiocc"

    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.InitDBInstances(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

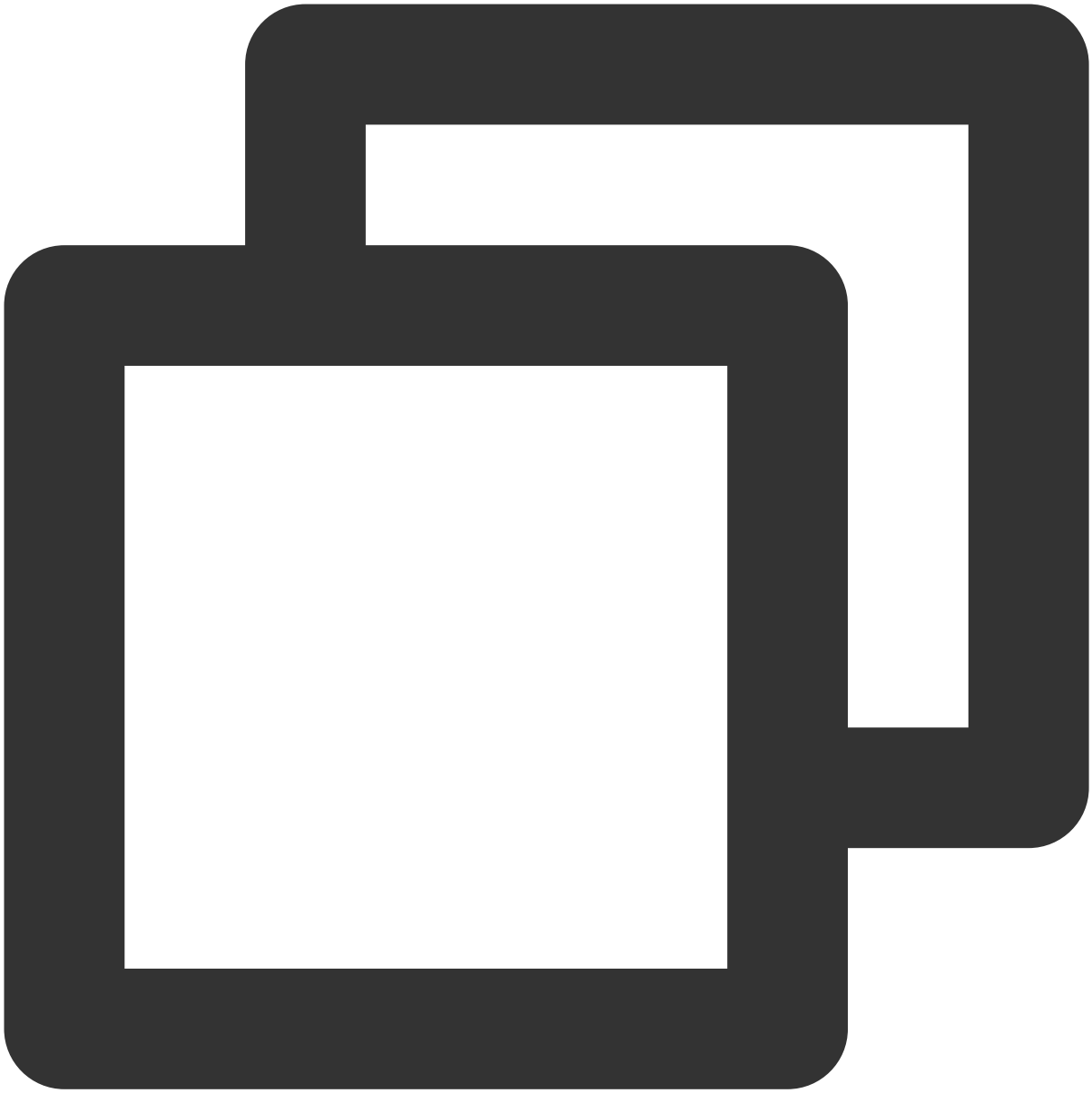
OpenDBInstanceGTID 인스턴스 GTID 활성화



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.OpenDBInstanceGTIDRequest()
    req.InstanceId = "cdb-7ghaiocc"


    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.OpenDBInstanceGTID(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

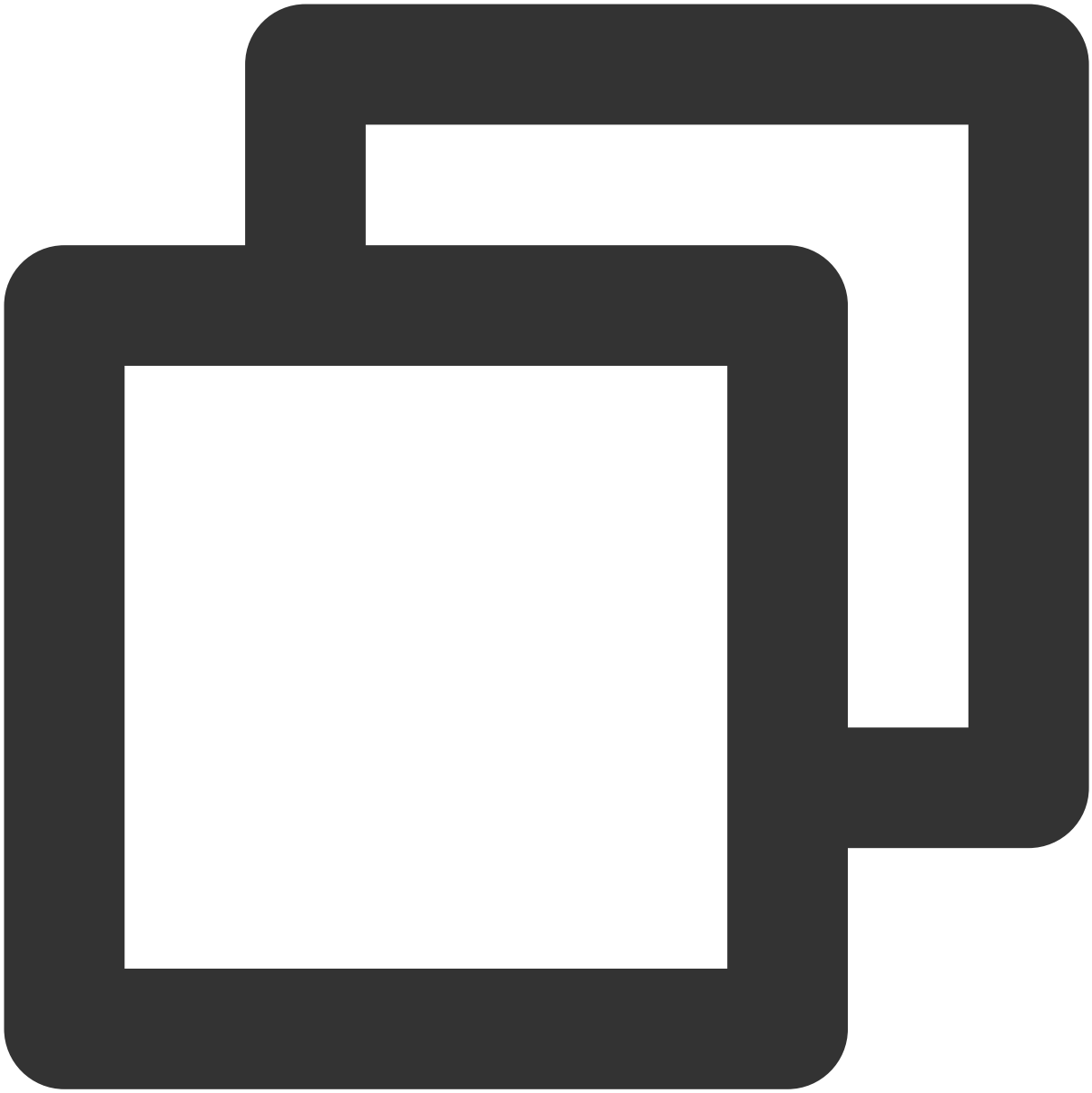
ModifyDBInstanceName CDB 인스턴스 이름 수정



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

# 클라우드 API 게이트 모듈 불러오기
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudS
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
    cred = credential.Credential("secretId", "secretKey")
```

```
    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-beijing")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.ModifyDBInstanceNameRequest()
    req.InstanceId = "cdb-cukm86n2"
    req.InstanceName = "1s 중국어”

    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.ModifyDBInstanceName(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

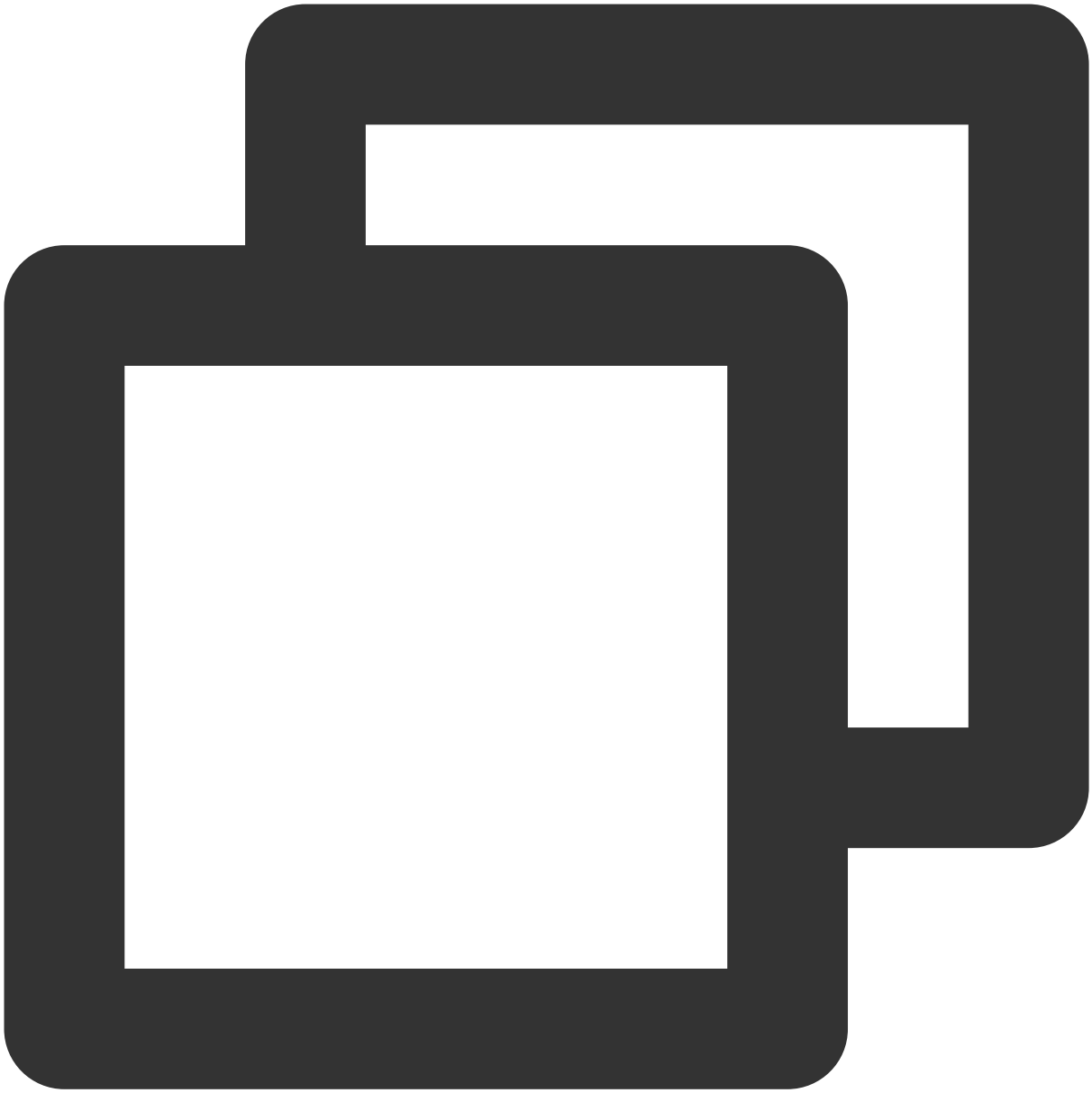
ModifyDBInstanceProject CDB 인스턴스 서브 프로젝트 수정



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

# 클라우드 API 게이트 모듈 불러오기
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:
        # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secr
        cred = credential.Credential("secretId", "secretKey")

        #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
        client = cdb_client.CdbClient(cred, "ap-shanghai")

        #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
        req = models.ModifyDBInstanceProjectRequest()
        req.InstanceIds = ["cdb-7ghaiocc"]
        req.NewProjectId =1


        # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
        resp = client.ModifyDBInstanceProject(req)

        # json 포맷의 문자열 출력
        print(resp.to_json_string())
    except TencentCloudSDKException as err:
        msg = traceback.format_exc() # 방식1
        print (msg)



DescribeDBInstancesList()
```

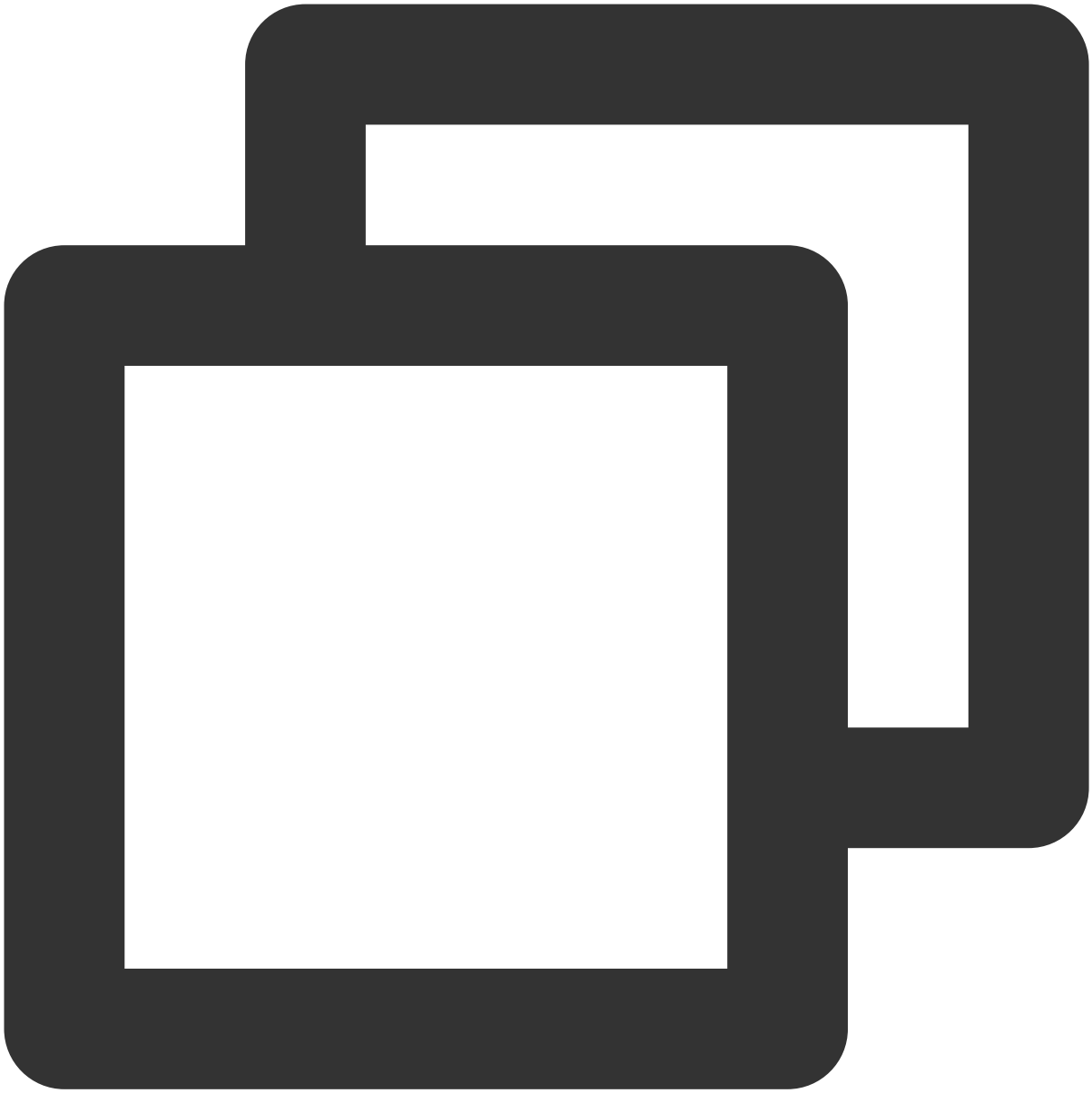
ModifyDBInstanceVipVport CDB 인스턴스 IP와 포트 번호 수정



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

# 클라우드 API 게이트 모듈 불러오기
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:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
    cred = credential.Credential("secretId", "secretKey")

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.ModifyDBInstanceVipVportRequest()
    req.InstanceId = "cdb-7ghaiocc"
    req.DstIp = "10.0.0.13"
    req.DstPort =1025
    req.UniqVpcId = 1111

    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.ModifyDBInstanceVipVport(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
        msg = traceback.format_exc() # 방식1
        print (msg)
```

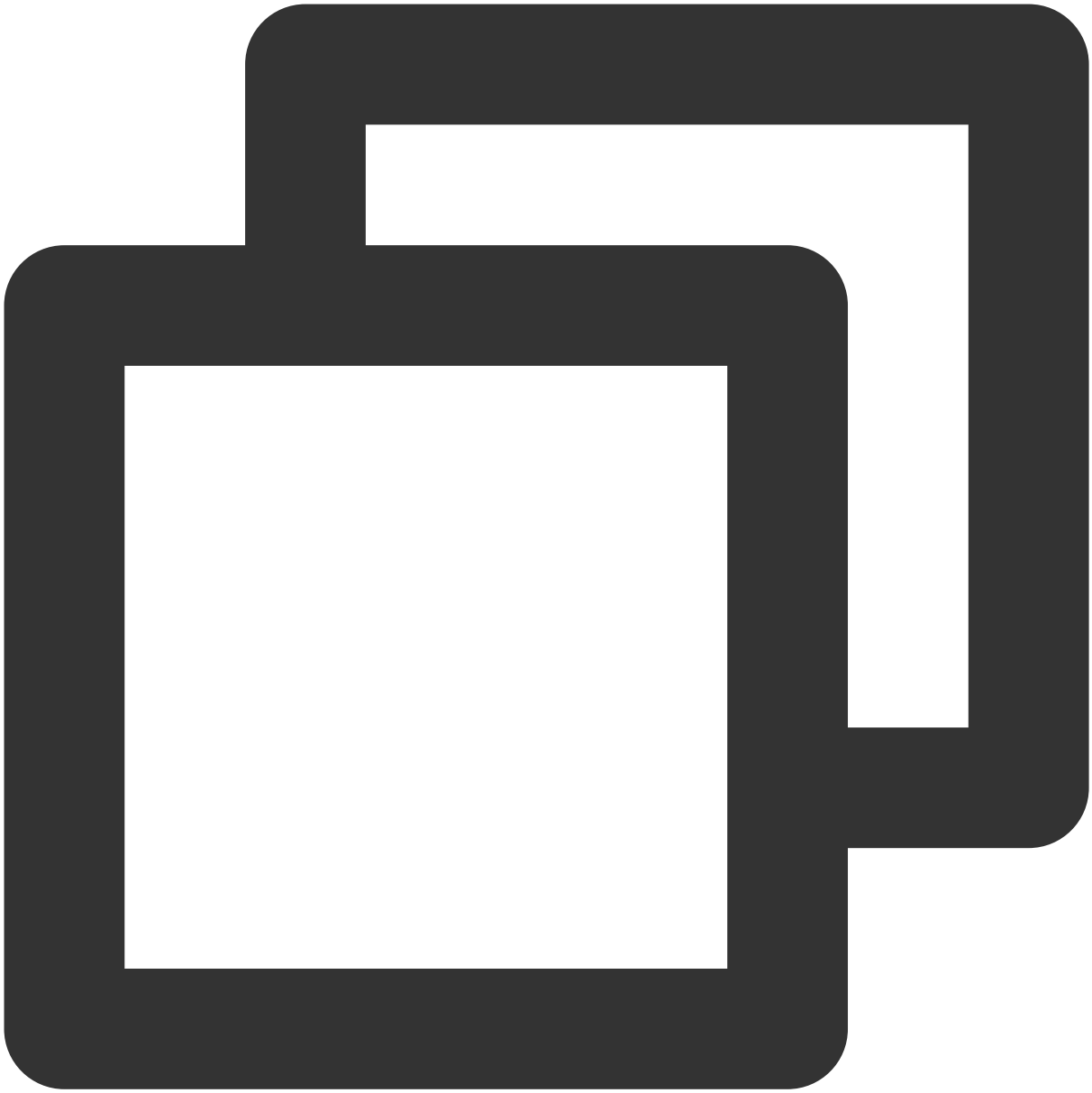
DescribeDBInstanceCharset CDB 인스턴스 문자 세트 조회



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBInstanceCharsetRequest()
    req.InstanceId = "cdb-1y6g3zj8"


    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.DescribeDBInstanceCharset(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

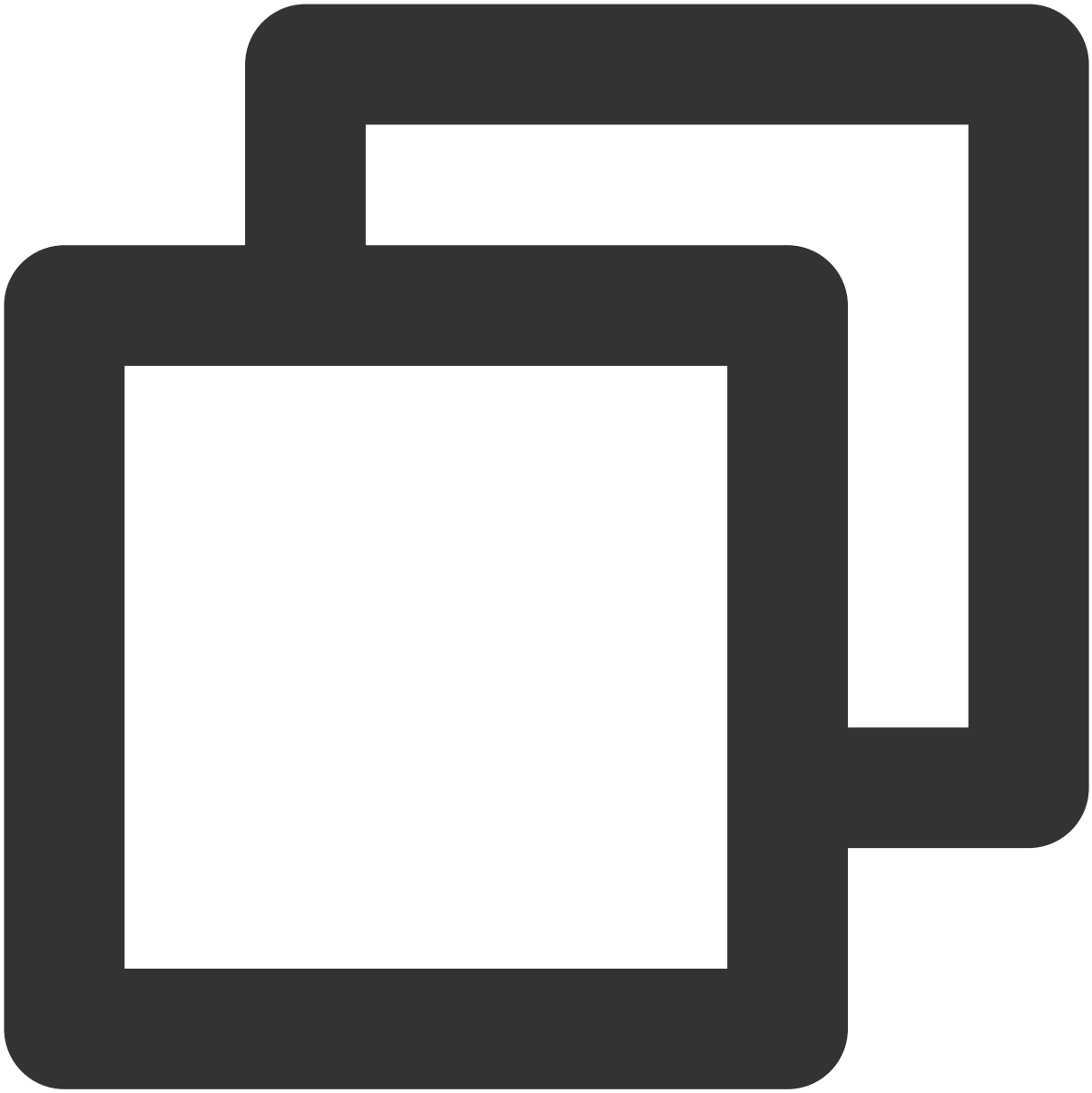
DescribeDBInstanceConfig CDB 인스턴스 설정 정보 조회



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBInstanceConfigRequest()
    req.InstanceId = "cdb-1y6g3zj8"


    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.DescribeDBInstanceConfig(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

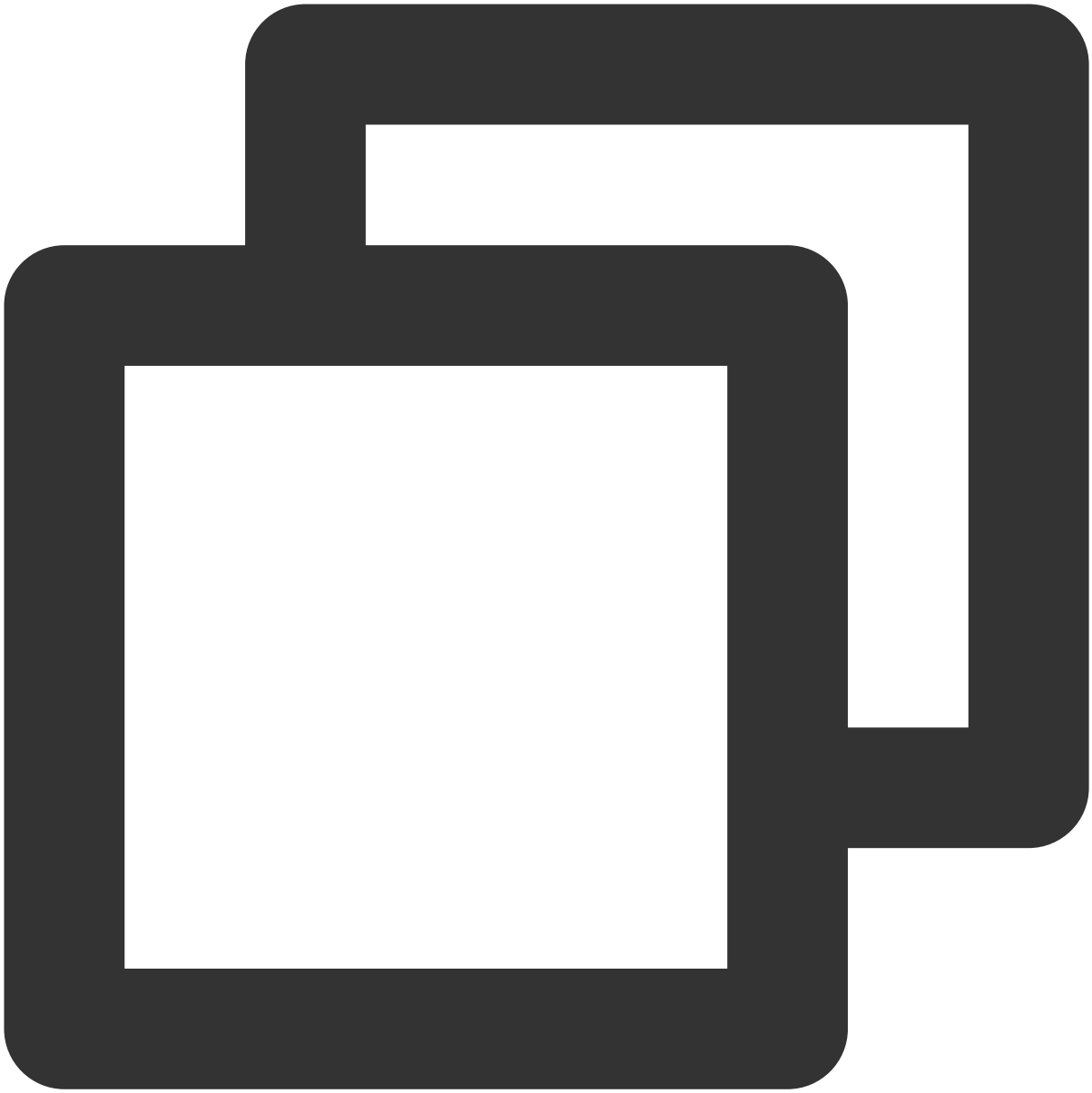
DescribeDBInstanceGTID CDB 인스턴스 GTID 활성화 여부 조회



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBInstanceGTIDRequest()
    req.InstanceId = "cdb-1y6g3zj8"


    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.DescribeDBInstanceGTID(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

DescribeDBInstanceRebootTime<42/>CDB 인스턴스 예상 재시작 시간 조회



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
```

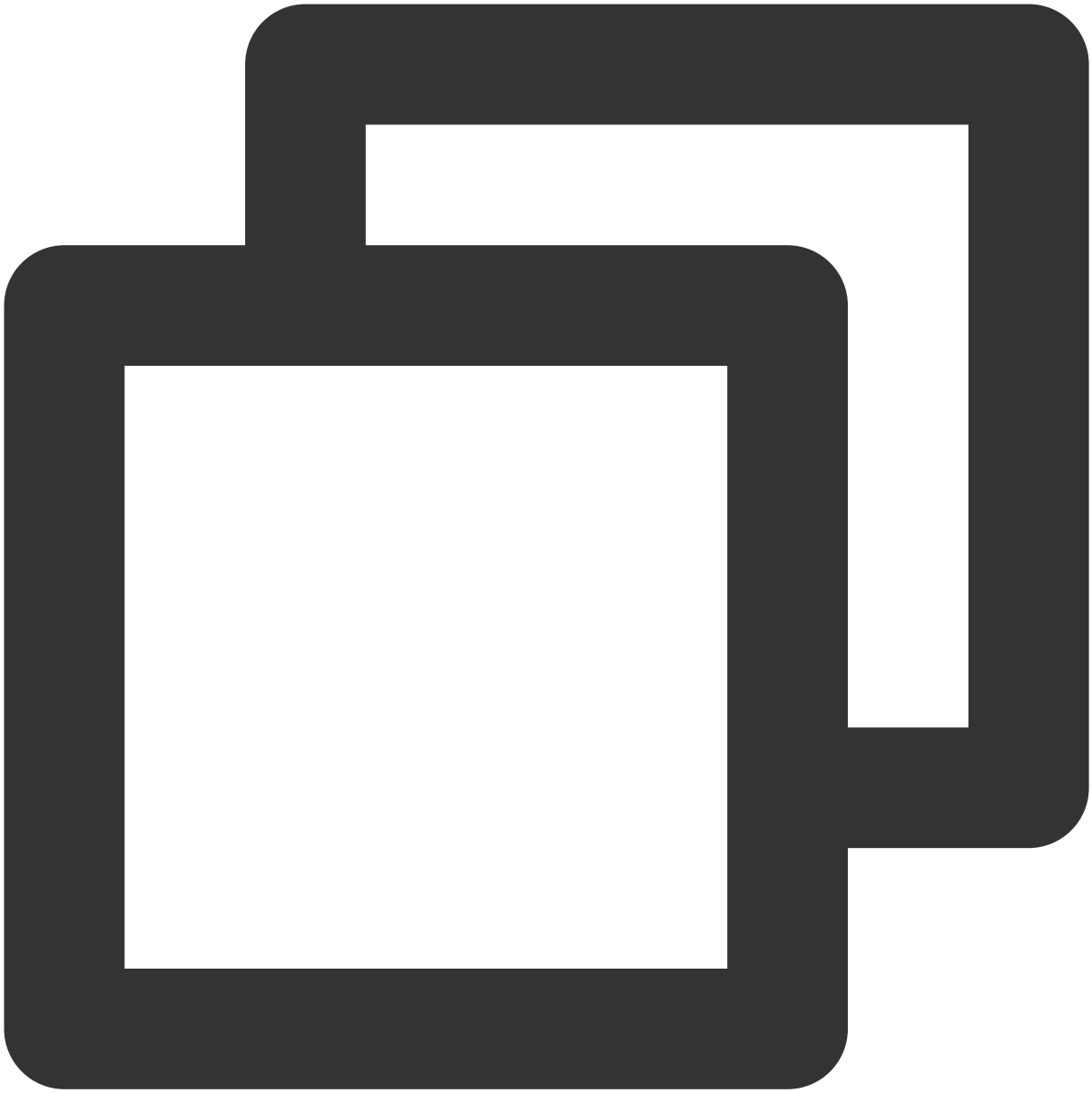
```
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
    cred = credential.Credential("secretId", "secretKey")

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBInstanceRebootTimeRequest()
    req.InstanceIds = ["cdb-1y6g3zj8"]


    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.DescribeDBInstanceRebootTime(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

백업 작업

최종 업데이트 날짜: : 2024-07-25 16:38:48

API	설명
CreateBackup	CDB 백업 생성
DeleteBackup	CDB 백업 삭제
DescribeBackupConfig	CDB 백업 설정 정보 조회
DescribeBackupDatabases	백업 데이터베이스 리스트 조회
DescribeBackupTables	지정 데이터베이스 백업 데이터 테이블 조회
DescribeBackups	백업 로그 조회
DescribeBinlogs	이진법 로그 조회
DescribeSlowLogs	슬로우 쿼리 로그 조회
ModifyBackupConfig	데이터베이스 백업 설정 수정

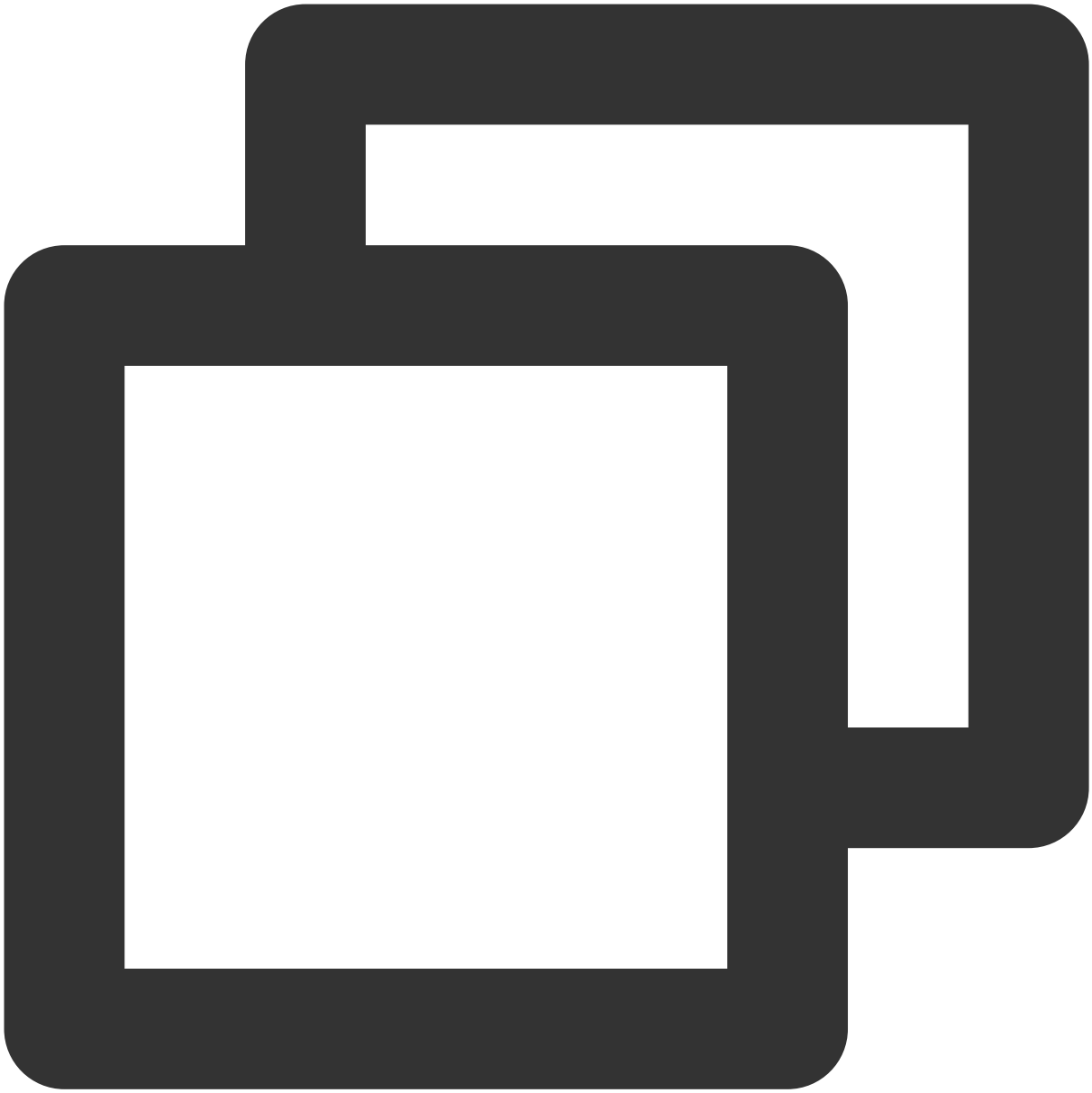
CreateBackup CDB 백업 생성



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.CreateBackupRequest()
    #req.MasterInstanceId = "cdb-7ghaiocc"
    req.BackupMethod = "logical"

    print req
    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.CreateBackup(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

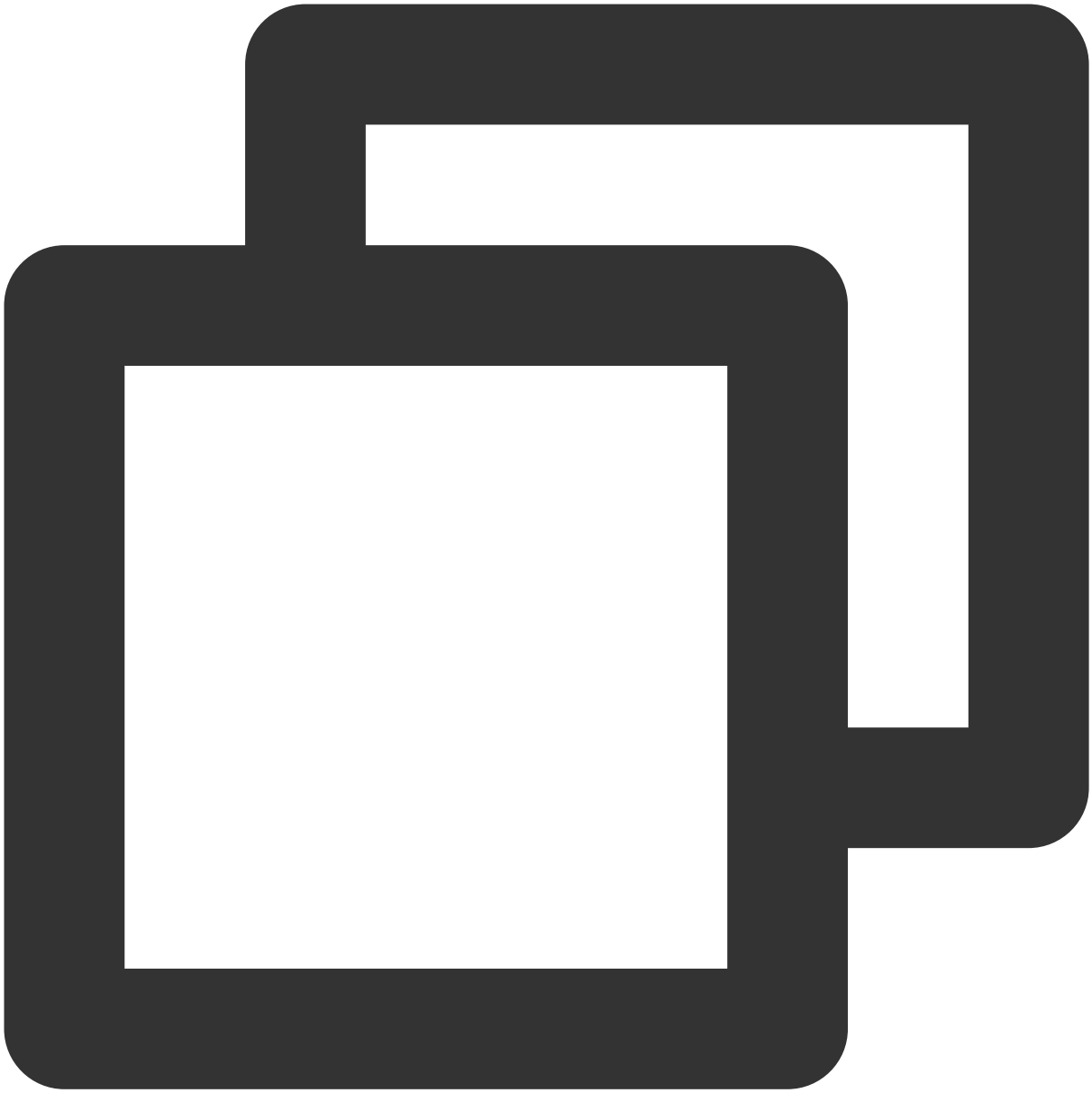
DeleteBackup CDB 백업 삭제



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")



    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.DeleteBackupRequest()

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


    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.DeleteBackup(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

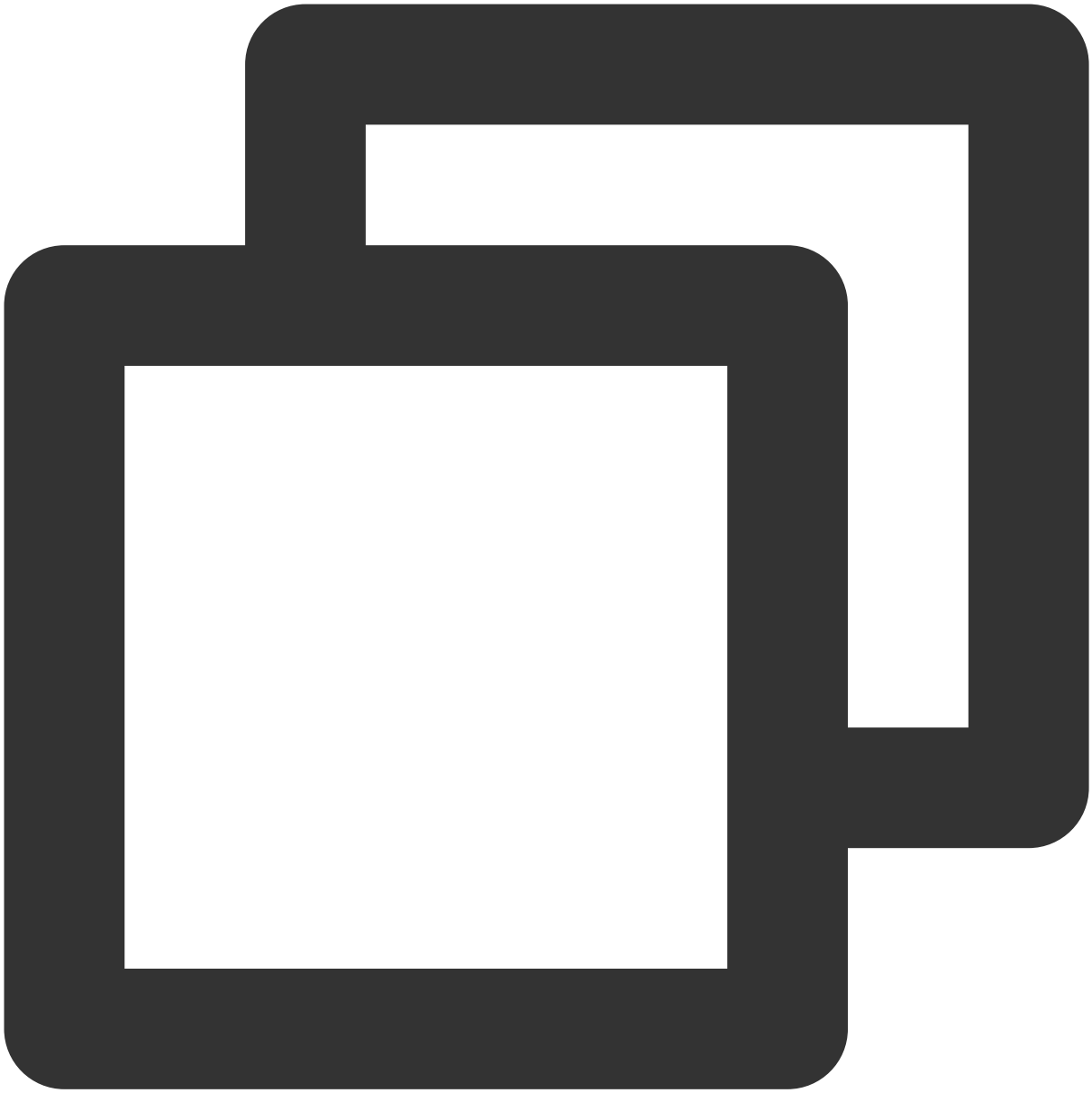
DescribeBackupConfig CDB 백업 설정 정보 조회



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBZoneConfigRequest()
    #req.MasterInstanceId = "cdb-7ghaiocc"

    print req
    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.DescribeDBZoneConfig(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

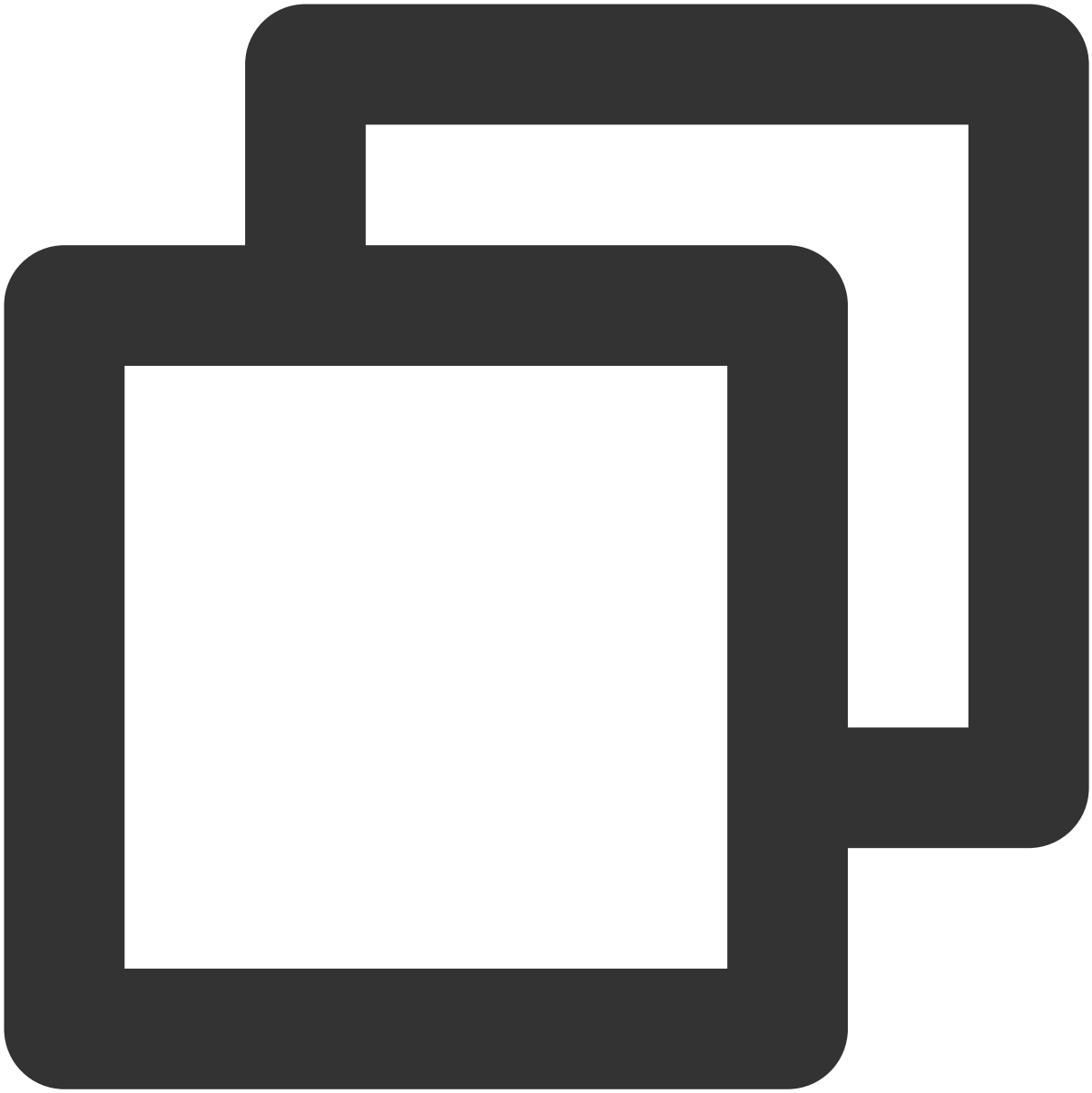
DescribeBackupDatabases 백업 데이터베이스 리스트 조회



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

# 클라우드 API 게이트 모듈 불러오기
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:
```

```
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
    cred = credential.Credential("secretId", "secretKey")

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.DescribeBackupDatabasesRequest()
    #req.MasterInstanceId = "cdb-7ghaiocc"
    req.StartTime = "2018-08-02 15:19:19"

    print req
    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.DescribeBackupDatabases(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    msg = traceback.format_exc() # 방식1
    print (msg)
```

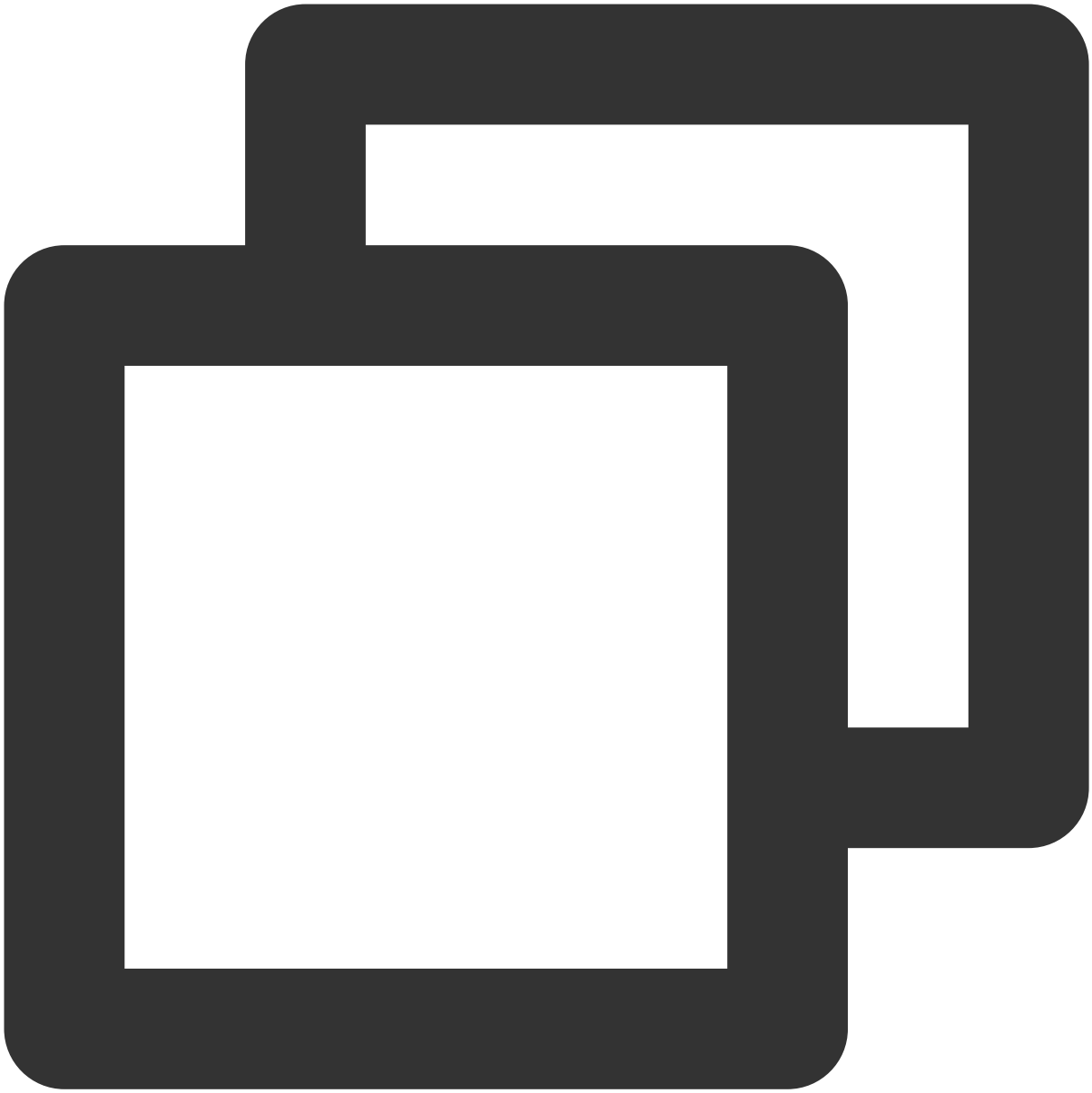
DescribeBackupTables 지정 데이터베이스 백업 데이터 테이블 조회



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.DescribeBackupTablesRequest()
    #req.MasterInstanceId = "cdb-7ghaiocc"
    req.StartTime = "2018-08-02 15:19:19"
    req.DatabaseName ="sissi"


    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.DescribeBackupTables(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

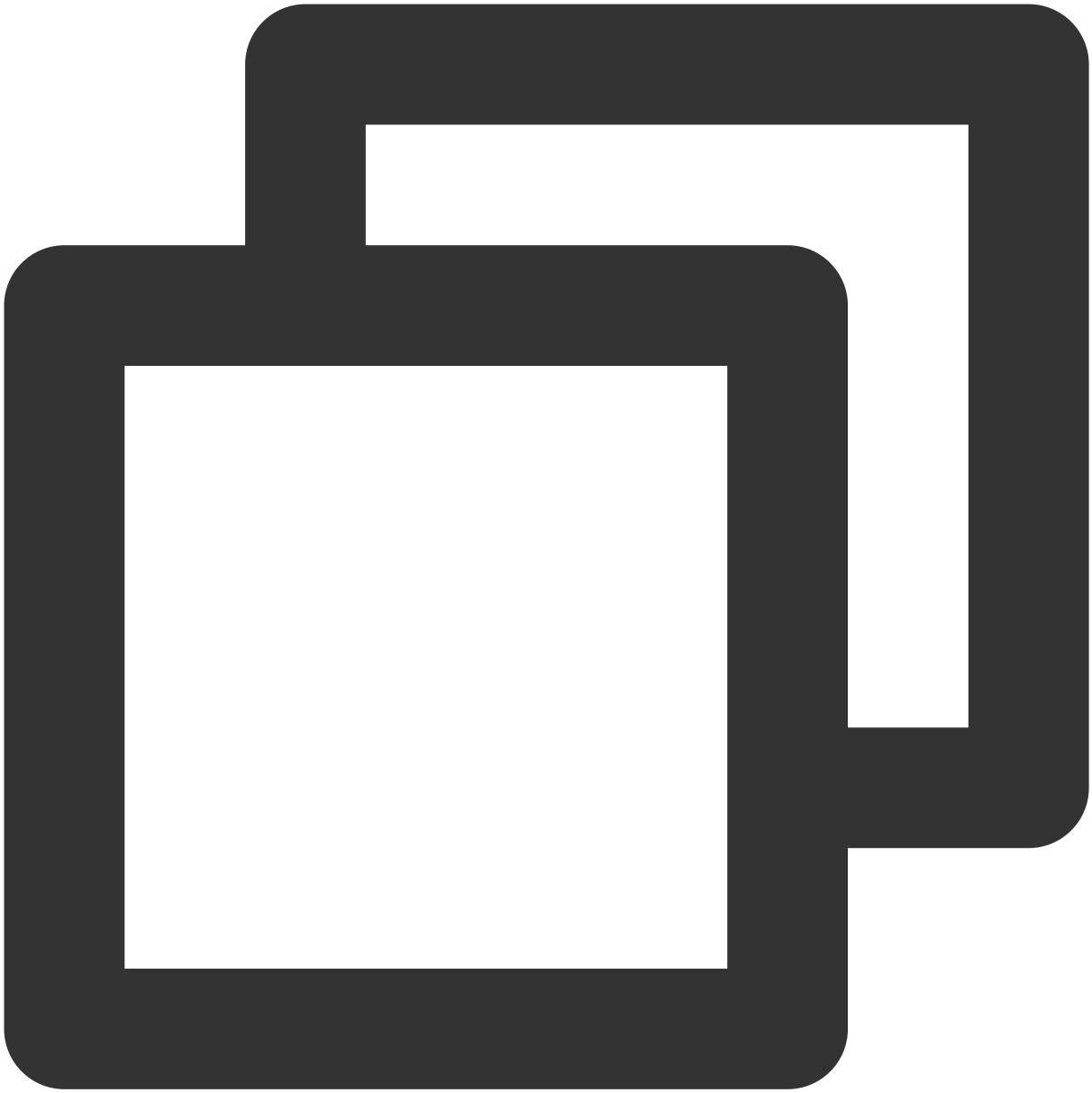
DescribeBackups 백업 로그 조회



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBPriceRequest()
    #req.MasterInstanceId = "cdb-7ghaiocc"



    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.DescribeBackups(req)
    print resp

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

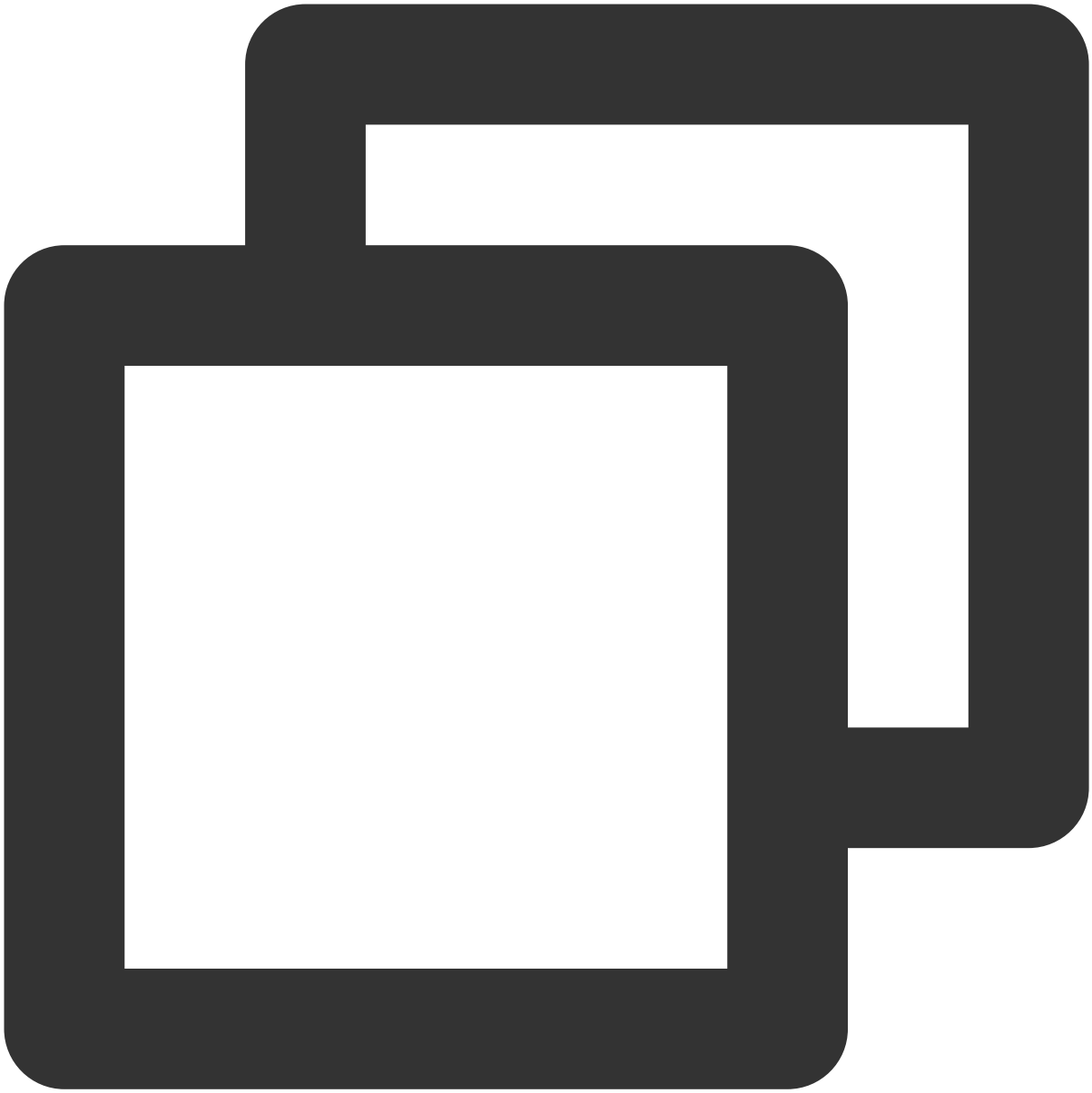
DescribeBinlogs 이진법 로그 조회



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.DescribeDBPriceRequest()
    #req.MasterInstanceId = "cdb-7ghaiocc"


    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.DescribeDBInstances(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

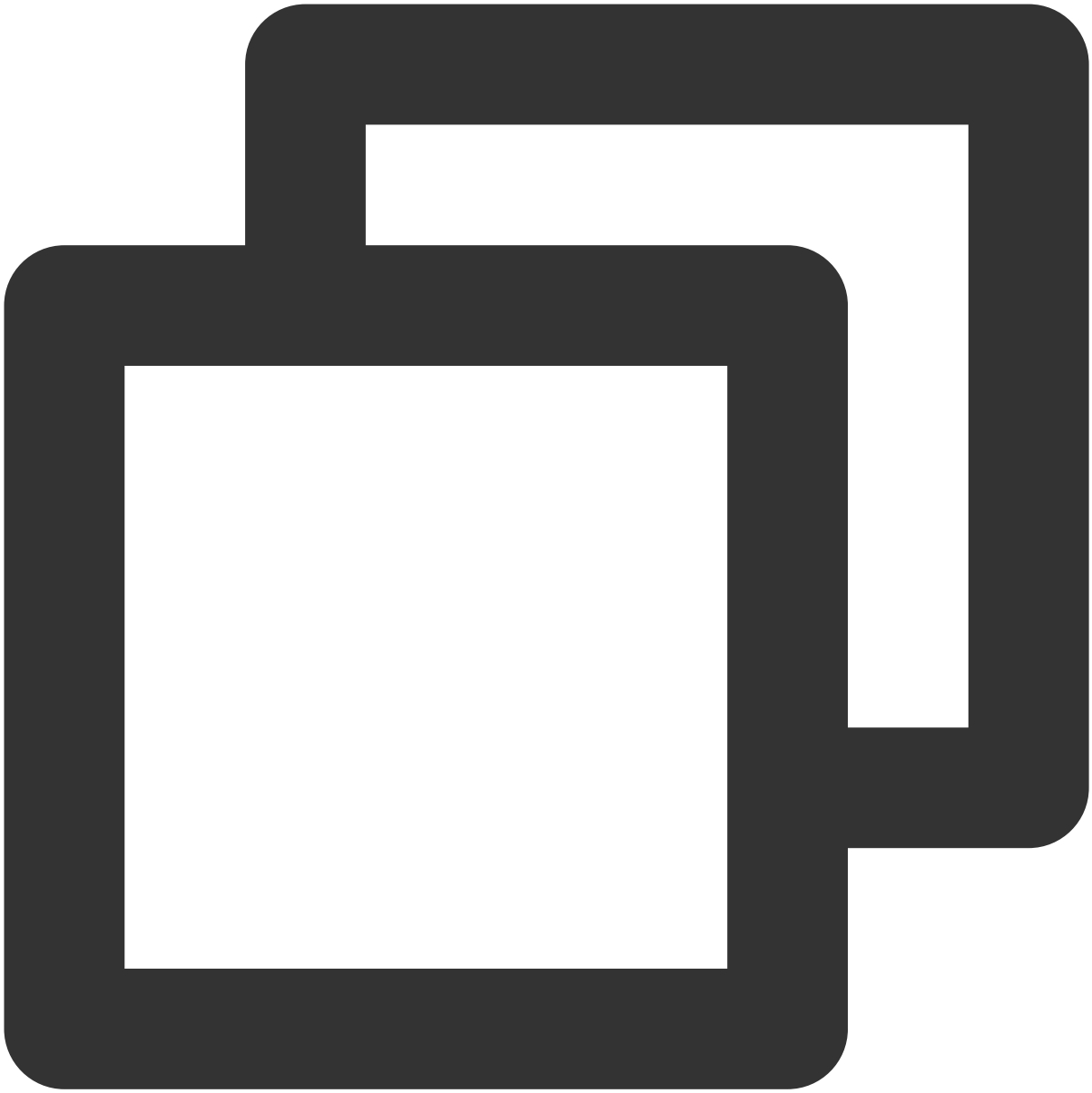
DescribeSlowLogs 슬로우 쿼리 로그 조회



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.DescribeSlowLogsRequest()
    #req.MasterInstanceId = "cdb-7ghaiocc"


    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.DescribeSlowLogs(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```

ModifyBackupConfig 데이터베이스 백업 설정 수정



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

# 클라우드 API 게이트 모듈 불러오기

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

try:
    # 1개의 인증 객체를 인스턴스화, 매개변수 입력 시 Tencent Cloud 계정 secretId, secretKe
```

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

    #인스턴스화 시 제품(예: cdb)의 client 객체 요청 필요
    client = cdb_client.CdbClient(cred, "ap-shanghai")

    #요청 객체 인스턴스화:req = models.ModifyInstanceParamRequest()
    req = models.ModifyBackupConfigRequest()
    req.InstanceId = "cdb-1y6g3zj8"
    req.ExpireDays = 10
    req.StartTime = "06:00-10:00"
    req.BackupMethod = "logical"
    print req


    # client 객체를 통해 액세스할 인터페이스 호출, 요청 객체 전송 필요
    resp = client.ModifyBackupConfig(req)

    # json 포맷의 문자열 출력
    print(resp.to_json_string())
except TencentCloudSDKException as err:
    print(err)
```