云数据库 MySQL

最佳实践

产品文档

腾讯云

文档目录

最佳实践
云数据库 MySQL 使用规范

最近更新时间：2018-09-18 17:11:45

目的

1. 规范化对云数据库 MySQL 的管理和维护，避免操作不当对云数据库 MySQL 造成不可用等影响；
2. 指导数据库开发人员合理编写 SQL，发挥云数据库 MySQL 最优性能。

受众

云数据库 MySQL 产品用户。

权限管理规范

1. 由于考虑到云数据库 MySQL 的稳定性和安全性，云数据库 MySQL 限制了 super、shutdown、file 权限，有时在云数据库 MySQL 上执行 set 语句时，会出现如下的报错：

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

   解决方法：如果需要 set 修改相关参数，可以使用控制台【参数修改】功能完成，如果需要修改的参数不在其中，可以提交工单后经评估后协助修改，确保实例稳定；

2. 按需授权，一般应用程序只授权 DML（SELECT、UPDATE、INSERT、DELETE）权限即可；
3. 授权对象最小化原则，一般的应用程序访问用户按库级别来授权；
4. 授权用户访问时只允许特定 IP 或 IP 段访问，可以在控制台配置安全组来做限制，安全组的设置请一定按照控制台提示的标准来操作，如果是公网访问设置安全组的场景，请一定放通所有涉及到的出口 IP；
5. 管理帐号与开发帐号分离。

日常操作规范

1. 尽量避免业务高峰期做 online ddl 操作，可以使用的工具请参考：`pt-online-sche-machange`；
2. 尽量避免业务高峰期批量操作数据，最好在业务低峰期分批来操作；
3. 尽量避免一个实例跑多个业务，耦合度太高会存在业务之间互相影响的风险；
4. 禁止使用弱密码，提升数据库实例安全性；
5. 建议关闭事务自动提交，线上操作养成 `begin;` 先行的习惯，降低误操作导致数据丢失的风险，误操作亦可使用 CDB 的回档功能（目前支持 5 天内任意时间点回档），若相关表不涉及跨库跨表的逻辑，可使用快速回档或者极速回档来更快恢复数据，回档新生成的库表名默认是原库表名 `_bak`；
6. 业务有推广活动等，请提前预估资源并做好实例相关优化，如需求量比较大时请及时与对应的服务经理联系；

7. 内网连接登录须确保 client 端的 CVM 机器与 CDB 是同一账号同一地域的机器；
8. 控制台下载的 binlog 日志，需要在本地解析的话，须确保客户端的 MySQL 版本与云数据库 MySQL 实例的版本一致，否则会出现解析出乱码的情况，建议使用 3.4 或以上版本的 mysqlbinlog；
9. 控制台上通过内网在 CVM 上下载冷备文件时，请用引号将 url 包起来，否则会出现 404 报错。

库表设计规范

1.通过业务场景分析和数据访问（包括数据库读写 QPS、 TPS、存储空间等）的预估，合理规划数据库使用资源，也可以在控制台云监控界面，配置云数据库 MySQL 实例的各项监控；

2.建库原则就是同一类业务的表放一个库，不同业务的表尽量避免公用同一个库，尽量避免在程序中执行跨库的关联操作，此操作对后续的快速回档也会产生一定的影响；

3.每张表必须要有主键，即使选不出合适的列做主键，亦必须添加一个无意义的列做主键 MySQL 第一范式标准 InnoDB 辅助索引叶子节点会保存一份主键值，推荐用自增短列作为主键，降低索引所占磁盘空间提升效率，`binlog_format` 为 row 的场景下，批量删数据没主键会导致严重的主从延迟；

4.认准 InnoDB 引擎 线上 CDB 产品，5.6 版本开始即不支持 MyISAM 引擎（官方 MySQL 8.0 开始也不支持），有 memory 引擎需求的，建议使用 redis 或者 memcached 自建数据库迁移到 CDB 会 MyISAM 自动转 InnoDB，故需满足以下两点否则会任务失败；

5.存在自增列的表，自增列上必须存在一个单独的索引，若在复合索引中，自增列必须置于第一位；

6. `row_format` 必须保证为非 fixed；

7.字符集统一使用 utf8mb4 降低乱码风险，部分复杂汉字和 emoji 表情必须使用 utf8mb4 方可正常显示，修改字符集只对修改后创建的表生效，故建议新购 CDB 初始化实例时即选择 utf8mb4；

8.小数字段推荐使用 decimal 类型，float 和 double 精度不够，特别是涉及金钱的业务，必须使用 decimal；

9.尽量避免据库中使用 text/blob 来存储大段文本、二进制数据、图片、文件等内容，而是将这些数据保存成本地磁盘文件，数据库中只保存其索引信息；

10.尽量不使用外键，建议在应用层实现外键的逻辑， 外键与级联更新不适合高并发场景，降低插入性能，大并发下容易产生死锁；

11.字段尽量定义为 NOT NULL 并加上默认值，NULL 会给 SQL 开发带来很多坑导致走不了索引，对 NULL 计算时只能用 IS NULL 和 IS NOT NULL 来判断；

12.降低业务逻辑和数据存储的耦合度，数据库存储数据为主，业务逻辑尽量通过应用层实现，尽可能减少对存储过程、触发器、函数、event、视图等高级功能的使用，这些功能移植性、可扩展性较差，若实例中存在此类对象，建议默认不要设置 definer，避免因迁移账号和 definer 不一致导致的迁移失败；

13.短期内业务达不到一个比较大的量级，建议禁止使用分区表。分区表主要用作归档管理，多用于快递行业和电商行业订单表，莫迷信分区表提升性能， 除非业务中 80% 以上的查询走分区字段；

14.对读压力较大，且一致性要求较低（接受数据秒级延时）的业务场景，建议购买只读实例从库来实现读写分离策略。

索引设计规范

1. 单表的索引数建议不超过 5 个，单个索引中的字段数建议不超过 5 个，太多就起不到过滤作用了，索引也占空间，管理起来也耗资源；
2. 选择业务中 SQL 过滤走的最多的并且 cardinality 值比较高的列建索引，业务 SQL 不走的列建索引是无意义的，字段的唯一性越高即代表 cardinality 值越高，索引过滤效果也越好，一般索引列的 cardinality 记录数小于 10% 我们可认为这是一个低效索引，例如性别字段；
3. varchar 字段上建索引时，建议指定索引长度，不要直接将整个列建索引， 一般 varchar 列比较长，指定一定长度作索引已经区分度够高，没必要整列建索引，整列建索引会显得比较重，增大了索引维护的代价，可以用 count(distinct left(列名, 索引长度))/count(*)来看索引区分度；
4. 避免冗余索引，两个索引(a,b) (a)同时存在，则(a)属于冗余索引——redundant index，若查询过滤条件为 a 列，(a,b)索引就够了，不用单独建(a)索引；
5. 建复合索引的时候，区分度最高的列放索引的在最左边， 例如 `select xxx where a = x and b = x;` ，a 和 b 一起建组合索引，a 的区分度更高，则建 `idx_ab(a,b)` 存在非等号和等号混合判断条件时，必须把等号条件的列前置。如： `where a xxx and b = xxx` 那么即使 a 的区分度更高，也必须把 b放在索引的最前列，因为走不到索引 a；
6. 禁止在更新十分频繁、区分度不高的列上建立索引，记录更新会变更 B+ 树，更新频繁的字段建立索引会大大降低数据库性能；
7. 合理利用覆盖索引来降低 io 开销，在 InnoDB 中二级索引的叶子节点保存的只保存本身的键值和主键值，若一个 SQL 查询的不是索引列或者主键，走这个索引就会先找到对应主键然后根据主键去找需要找的列，这就是回表，这样会带来额外的 io 开销，此时我们可以利用覆盖索引来解决这个问题，比如 `select a,b from xxx where a = xxx` ，若 a 不是主键，这时候我们可以创建 a，b 两个列的复合索引，这样就不会回表。

SQL编写规范

1. 按需索取，拒绝 `select *` ，规避以下问题：
   无法索引覆盖，回表操作，增加 io；
   额外的内存负担，大量冷数据灌入 `innodb_buffer_pool_size` ，降低查询命中率；
   额外的网络传输开销；
2. UPDATE、 DELETE 操作不使用 LIMIT，必须走 WHERE 精准匹配，limit 是随机的，此类操作会导致数据出错；
3. 禁止使用 `INSERT INTO t_xxx VALUES(xxx)` ，必须显示指定插入的列属性，避免表结构变动导致数据出错；
4. 尽量避免使用大事务，建议大事务拆小事务，规避主从延迟；
5. 业务代码中事务及时提交，避免产生没必要的锁等待；
6. 少用多表 join，大表禁止 join，两张表 join 必须让小表做驱动表，join 列必须字符集一致并且都建有索引；
7. LIMIT 分页优化，LIMIT 80000，10 这种操作是取出 80010 条记录，再返回后 10 条，数据库压力很大，推荐先确认首记录的位置再分页，如： `SELECT * FROM test WHERE id = ( SELECT sql_no_cache id FROM test order by id LIMIT 80000,1 ) LIMIT 10；` ；

8. 避免多层子查询嵌套的 SQL 语句 MySQL5.5 之前的查询优化器把会把 in 改成 exists，走不到索引性，若外表很大则性能会很差；
9. SQL 语句中最常见的导致索引失效的情况需注意：
   - 隐式类型转换，如索引 a 的类型是 varchar，SQL 语句写成 where a = 1;; varchar 变成了int ；
   - 对索引列进行数学计算和函数等操作，例如，使用函数对日期列进行格式化处理；
   - join 列字符集不统一；
   - 多列排序顺序不一致问题，如索引是 (a,b)，SQL 语句是 order by a b desclike；
   - 模糊查询使用的时候对于字符型 `xxx%` 形式可以走到一些索引，其他情况都走不到索引；
   - 使用了负方向查询（not，!=，not in 等）。

TIPS：

1. 上述情况很难完全避免，推荐方案是不要将此类条件作为主要过滤条件，跟在走索引的主要过滤条件之后则问题不大；
2. 监控上发现全表扫描的量比较大，可以在控制台参数设置 `log_queries_not_using_indexes` ，稍后下载慢日志文件分析，但不要开太久以免慢日志暴增；
3. 业务上线之前做有必要的 SQL 审核，日常运维需定期下载慢查询日志做针对性优化。

为云数据库 MySQL创建VPC

最近更新时间：2018-09-18 17:11:37

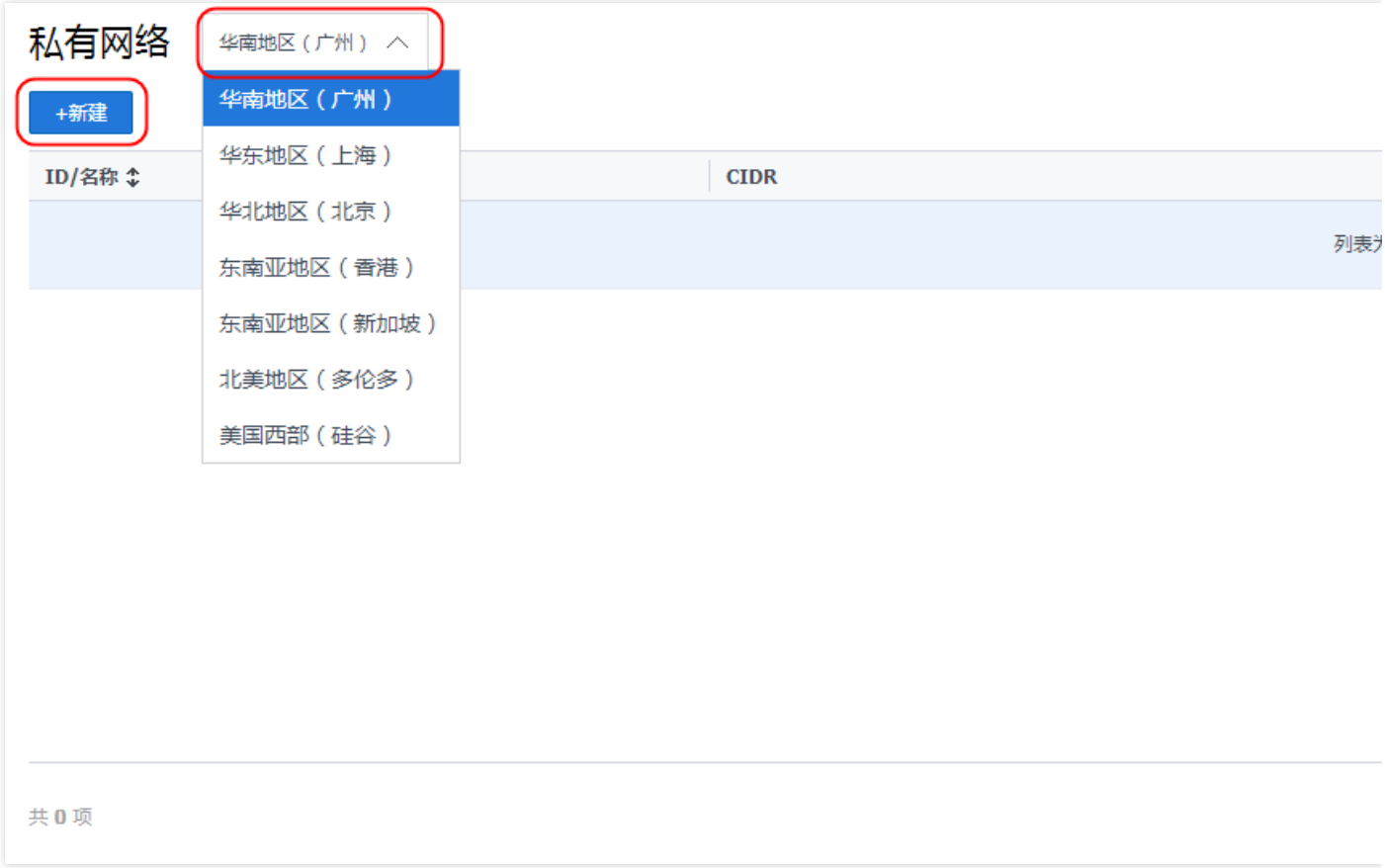
腾讯云提供托管腾讯云 CDB 数据库的平台：腾讯云 VPC。利用腾讯云 VPC ，您可以在 VPC 中启动腾讯云资源，如腾讯云 CDB 数据库实例。VPC 产品详细说明请参照 私有网络。
一种常见的方案是运行在同一 VPC 的腾讯云 CDB 数据库实例和 Web 服务器共享数据。在本教程中，针对此方案创建 VPC，并将云数据库添加进 VPC 以配合使用。

步骤一：创建私有网络、初始化子网和路由表

私有网络至少包含一个子网，只有在子网中才可以添加云服务资源。

1. 登录 腾讯云控制台，单击导航条【云产品】 > 【基础产品】 > 【云计算与网络】 > 【私有网络】，或者单击腾讯云 私有网络介绍页中 的【立即使用】按钮，进入 私有网络控制台。

2. 选择列表上方下拉框中的地域，单击【新建】创建私有网络，例如，选择地域：华南地区（广州）。



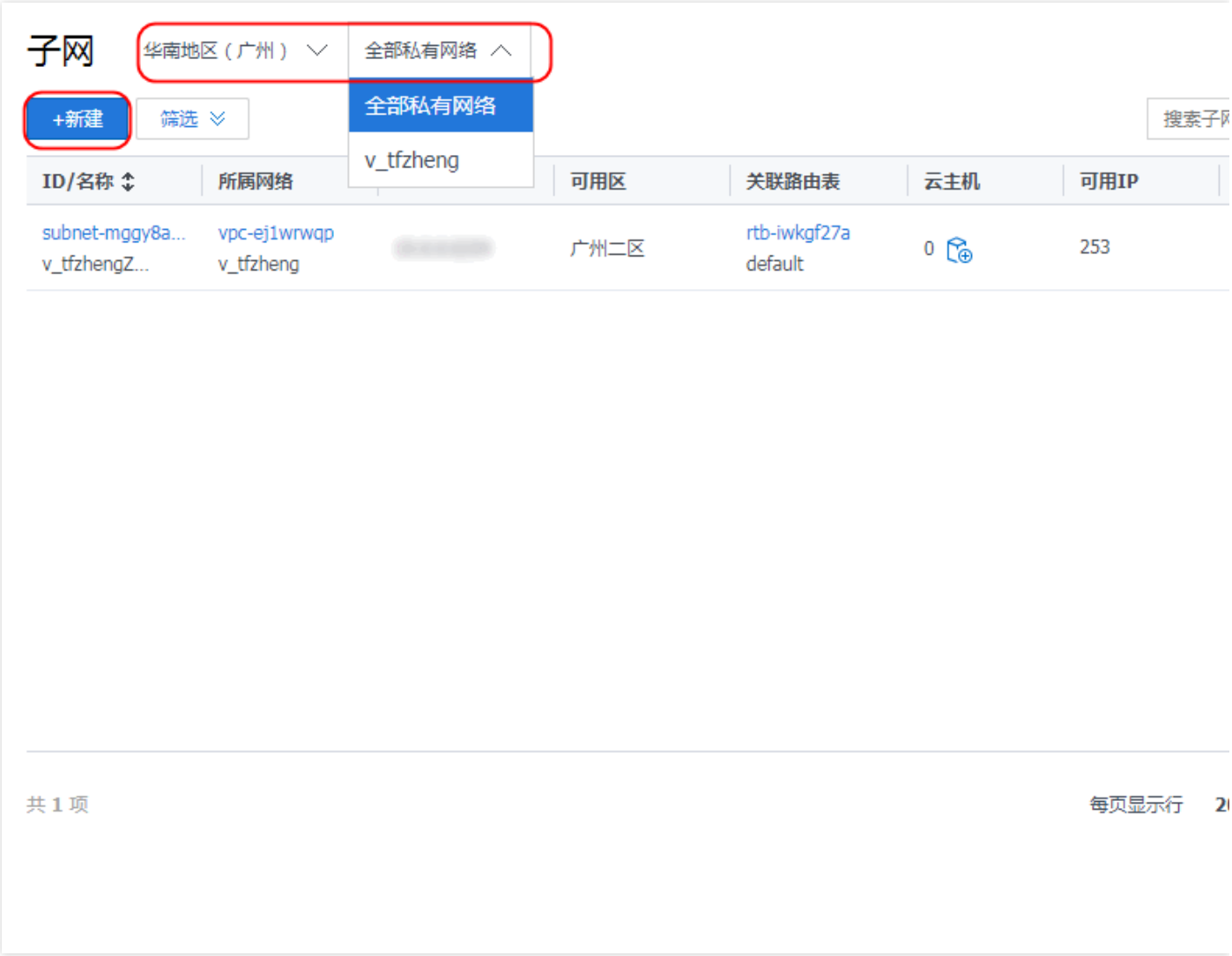
3. 填写私有网络和子网的名称和 CIDR，并选择子网的可用区，单击【创建】。

步骤二：创建子网

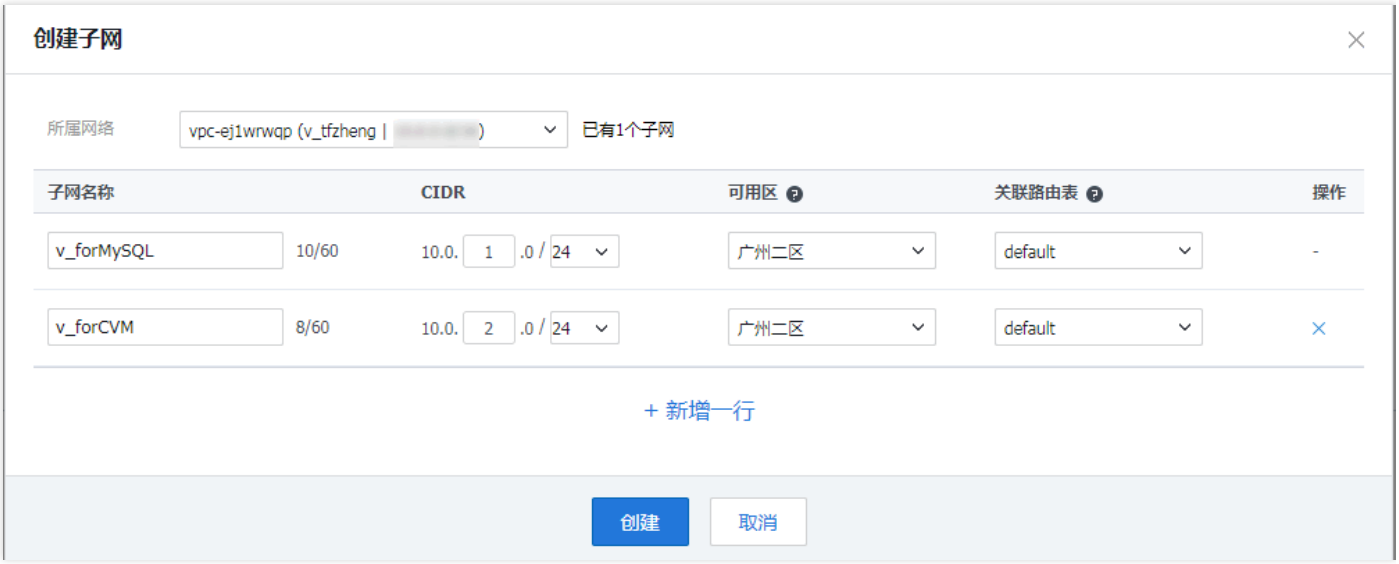
您可以同时创建一个或多个子网。

1. 在 私有网络控制台 中，单击左导航栏中的【子网】。

2. 选择下拉框中的地域和私有网络，单击【新建】。



3. 填写子网络名称、CIDR、可用区和关联路由表，单击【创建】。单击【新增一行】，可以同时创建多个子网。

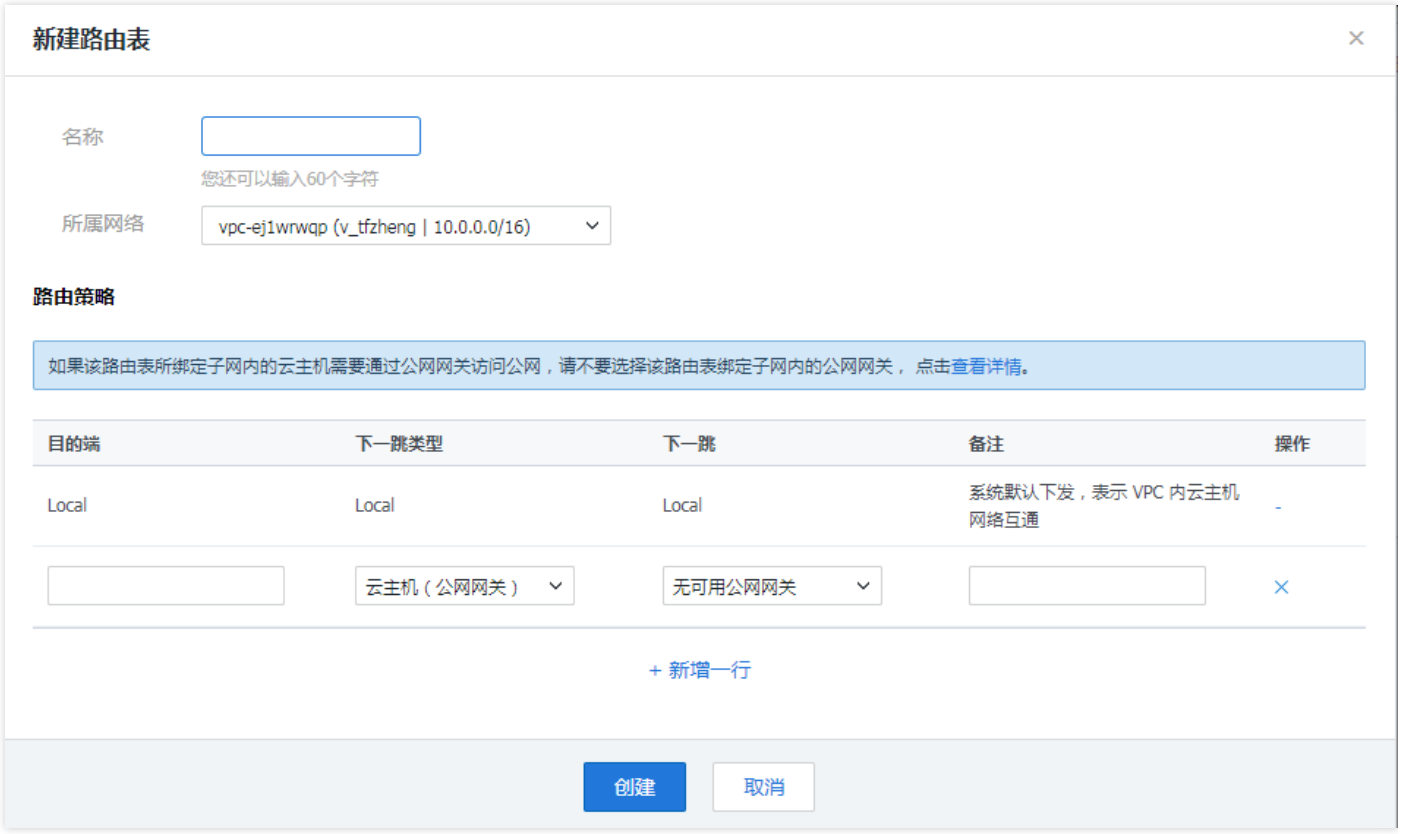


步骤三：新建路由表关联子网

您可以创建自定义路由表、编辑路由策略、然后关联指定子网，子网关联的路由表用于指定该子网的出站路由。

1. 在 私有网络控制台 中，单击左导航栏中的【路由表】，选择下拉框中的地域和私有网络，单击【新建】。

2. 在创建路由表弹出框中输入名称、所属网络及新建路由策略。单击【创建】，即可在路由表列表中看到您新建的路由表。



3. 单击左导航栏中的【子网】。鼠标移动到需要关联该路由表的【子网】一行， 编辑按钮即会出现在【关联路由表】列中。单击【编辑按钮】，在下拉框中选择关联路由表。单击【保存】。

步骤四：添加云数据库

新购的云数据库支持在私有网络中使用，需要注意的是，网络一旦选定将不可更改。

1. 登录腾讯云 管理控制台，将鼠标移至导航条中的【云产品】>【基础产品】>【数据库】单击【关系型数据库】，进入 云数据库控制台，单击【新建】按钮，进入云数据库 MySQL 购买界面。

2. 在云数据库选购页【网络】选项，单击私有网络，选择之前创建的私有网络以及相应子网，将新购的云数据库添加进私有网络。

步骤五：添加云服务器

新购的云服务器支持在私有网络中使用，需要注意的是，**网络一旦选定将不可更改。**

进入云服务器 产品介绍页，单击【立即选购】后，在产品购买页的【网络类型】选择【私有网络】，选择与之前数据库相同的 VPC，将新购的云服务器添加到与云数据库相同的 VPC 内。

使用云数据库 MySQL 提高业务负载能力

最近更新时间：2018-10-11 09:33:31

具备优异性能和扩展能力的数据库可帮助您迅速提高原有系统的负载能力，在同等数据库规模下合理使用 TencentDB 可帮助您提高数据库的并发能力，支撑更高的业务每秒访问次数。

1. 选择适合自己的数据库配置

1.1 选择数据库版本

在腾讯云上TencentDB for MySQL目前提供完全兼容原生MySQL的5.5，5.6，5.7版本供选择，我们强烈建议您选择5.6或更高的版本，它们提供了更稳定的数据库内核，优化改进5.5及更老版本的设计以提升系统的性能，并提供了多项极具吸引力的新特性。我们将以MySQL 5.7为例介绍新版本特性。

MySQL 5.7是 TencentDB 目前提供的最新版本，具有被普遍认可的高性能、可靠性和易用性。它部分优化点和新特性如下：

原生 JSON 支持

在 MySQL 5.7 中，新增了一种新的数据类型，用来在 MySQL 的表中存储 JSON 格式的数据。原生支持 JSON 数据类型主要有如下好处:

• 文档校验 – 只有符合JSON规范数据段的才能被写入类型为 JSON 的列 中，所以其实相当于有了自动JSON语法校验。

• 高效访问 – 更重要的是，当您在一个 JSON 类型的列中存储 JSON 文档的时候，数据不会被视为纯文本进行存储。实际上，数据用一种优化后的二进制格式进行存储，以便可以更快速地访问其对象成员和数组元素。

• 性能 – 可以在 JSON 类型列的数据上创建索引以提升 query 性能。这种索引 可以由在虚拟列上所建的"函数索引"来实现。

• 便捷 – 针对 JSON 类型列附加的内联语法可以非常自然地在 SQL 语句中集 成文档查询。例如(features.feature 是一个 JSON 字段):

```
SELECT feature->"$.properties.STREET" AS property_street FROM features WHERE id = 121254;
```

使用 MySQL 5.7，您现在可以在一个工具中无缝地混合最好的关系和文档范例，在 不同的应用和使用案例中应用关系型范例或文档性范例当中最适合的范例。这为 MySQL 用户大大扩大了应用范围。

Sys Schema

MySQL SYS Schema 是一个由一系列对象(视图、存储过程、存储方法、表和触 发器)组成的 database schema，以使主要存储在 Performance Schema 和 INFORMATION_SCHEMA 中诸多的表中监测数据资源可以以方便、可读、对 DBA 和开发者的友好的方式进行访问。

MySQL SYS Schema 默认包含在 MySQL 5.7 中，并提供摘要视图以回答诸如下面所列的常见问题:

• “谁占了我数据库服务的所有资源?”

• “哪些主机对数据库服务器的访问量最大?”

• “我实例上的内存都上哪去了?”

InnoDB相关改进

- InnoDB 在线操作（Online DDL）：现在您可以在不重启 MySQL 的情况下，动态地调整您的 Buffer Pool size 以适应需求的 改变。现在 InnoDB 也可以在线自动清空 InnoDB 的 UNDO 日志和表空间，这样就消除了产生大共享表空间文件 (ibdata1) 问题的一个常见原因。最后，MySQL 5.7支持重命名索引和修改varchar的大小，这两项操作在之前的版本中，都需要重建索引或表。
- InnoDB 原生分区：在 MySQL 5.7 InnoDB 中包含了对分区的原生支持。InnoDB 原生分区会降低负载，减少多达 90% 的内存需求。
- InnoDB 缓存预热：现在，当 MySQL 重启时，InnoDB 自动保留您缓存池中最热的 25% 的数据。您再也不需要任何预加载或预热您数据缓存的工作，也不需要承担 MySQL 重启带来的性能损失。

MySQL 5.7 更多优化和新特性可查看MySQL官方资料

1.2 选择实例规格(数据库内存)

当前 TencentDB for MySQL并未提供单独的CPU选项，CPU将根据内存规格按比例分配。您可以根据自己的业务特征购买相应的数据库规格，我们为每一种实例都做了详尽的标准化测试以为您提供选型时的性能参考。但需要注意的是，Sysbench标准化测试并不能代表所有的业务场景，建议您在将业务正式运行在 TencentDB 之前对数据库做一次压力测试，以便于更加了解 TencentDB 在您的业务场景下的性能表现。

查看 TencentDB for MySQL性能说明

内存是实例的核心指标之一，访问速度远远大于磁盘。通常情况下，内存中缓存的数据越多，数据库的响应就越快；如果内存较小，当数据超过一定量后，就会被刷新到磁盘上，如果新的请求再次访问该数据，就要从磁盘上把它从磁盘中读取进内存，消耗磁盘io，这个时候数据库响应就会变慢。

对于读并发较大或读延迟较为敏感的业务，我们建议您不要选择过小的内存规格，以保障数据库的性能。

1.3 选择硬盘（数据存储空间）

TencentDB 数据库实例的硬盘空间仅包括MySQL数据目录，不含binlog、relaylog、undolog、errorlog、slowlog日志空间。在写入的数据量超出实例硬盘空间时，如未及时升级，可能会触发实例锁定。因此在选购硬盘空间时，建议您对未来一段时间内可能的数据量增长保留一定冗余，避免因硬盘容量不足引起的实例锁定或频繁升级。

1.4 选择适合您的数据复制方式

TencentDB for MySQL提供了异步、半同步、强同步三种复制方式。了解 TencentDB 的数据复制方式

如您的业务对写入时延或数据库性能较为敏感，建议您选择异步复制方式。

1.5 云数据库的高可用

TencentDB for MySQL采用的主备M-M的高可用架构，其主备之间的数据同步依靠binlog日志的方式。同时支持将实例恢复到任何一个时间点，这个功能需要依靠运用备份和日志。因此，通常情况下您无需再搭建备份恢复系统或付出其他额外支出来保障实例的高可用。

1.6 云数据库的扩展性

TencentDB for MySQL的数据库版本，内存/硬盘规格均支持在线的动态热升级。升级过程不会中断您的业务，您无需担心业务规模增长带来的数据库瓶颈。

1.7 将CVM和CDB配合使用

通常情况下，在购买成功后您需要将CVM和TencentDB配合使用。
查看如何使用云主机CVM访问TencentDB

2. 使用只读实例作为读扩展

在常见的互联网业务中，数据库读写比例通常为4：1至10：1之间。在这类业务场景下，数据库的读负载远高于写负载，在遇到性能瓶颈时一个常见的解决方案就是增加读负载。

TencentDB for MySQL 只读实例为您提供了读扩展解决方案。了解如何使用只读实例

只读实例也可应用于不同业务的只读访问中，如：主实例承担在线业务读写访问，只读实例为内部业务或数据分析平台提供只读查询。

3. 云数据库的灾备方案

TencentDB for MySQL 提供了灾备实例帮助您一键搭建跨城域的异地数据库灾备。

使用灾备实例，可实现多地域之间不同机房互为冗余，当一个机房发生故障或由于不可抗因素导致无法提供服务，快速切换到另外一个机房。灾备实例使用腾讯内网专线做数据同步，且经过TencentDB的内核级复制优化，尽可能消除灾难情况下同步延迟给业务带来的影响，在异地业务逻辑就绪的情况下，可以达到秒级别的灾备切换。

4. 两地三中心方案

使用TencentDB，仅需页面简单配置几步即可实现两地三中心方案：

- 购买 TencentDB 同城强一致性集群，选择多可用区部署（灰度开放），提供一地两中心能力。
- 为该集群添加异地灾备节点，即可实现两地三中心架构。

5. 使用灾备实例为用户提供就近接入

灾备实例同样具备主备M-M的高可用架构，同时可对外提供只读的访问能力。因此，如您有需要跨地域的用户就近接入业务场景，您可放心使用灾备实例。

部署 Python Web 应用程序

最近更新时间：2018-07-27 17:35:24

Django是一个开放源代码的Web应用框架，由Python写成。
本教程指导如何部署默认 Django 网站至运行 Python 2.7 的 CVM 云服务器。

使用的软件环境为：CentOS7.2 | Python2.7 | Django1.11。

登录到云服务器实例

云服务器的购买和访问请参考 快速入门 Linux 云服务器。

安装 Python

在 CentOS 中会会默认安装 python，您可通过 `python --version` 查看 python 版本。

安装 Django

1. 安装 pip。

```
yum install python-pip -y
```

2. 通过 pip 安装 Django。

```
pip install Django
```

3. 查看 Django 版本，测试 Django 是否安装完成。

```
python # 进入python命令行
>>> import django
>>> django.VERSION
```

安装 MySQLdb 模块

安装 MySQL 的支持模块。

```
yum install python-devel
yum install mysql-devel
pip install MySQL-python
```

安装 Apache 服务

1. 在云服务器实例中使用 `yum` 安装 Apache。

```
yum install httpd -y
```

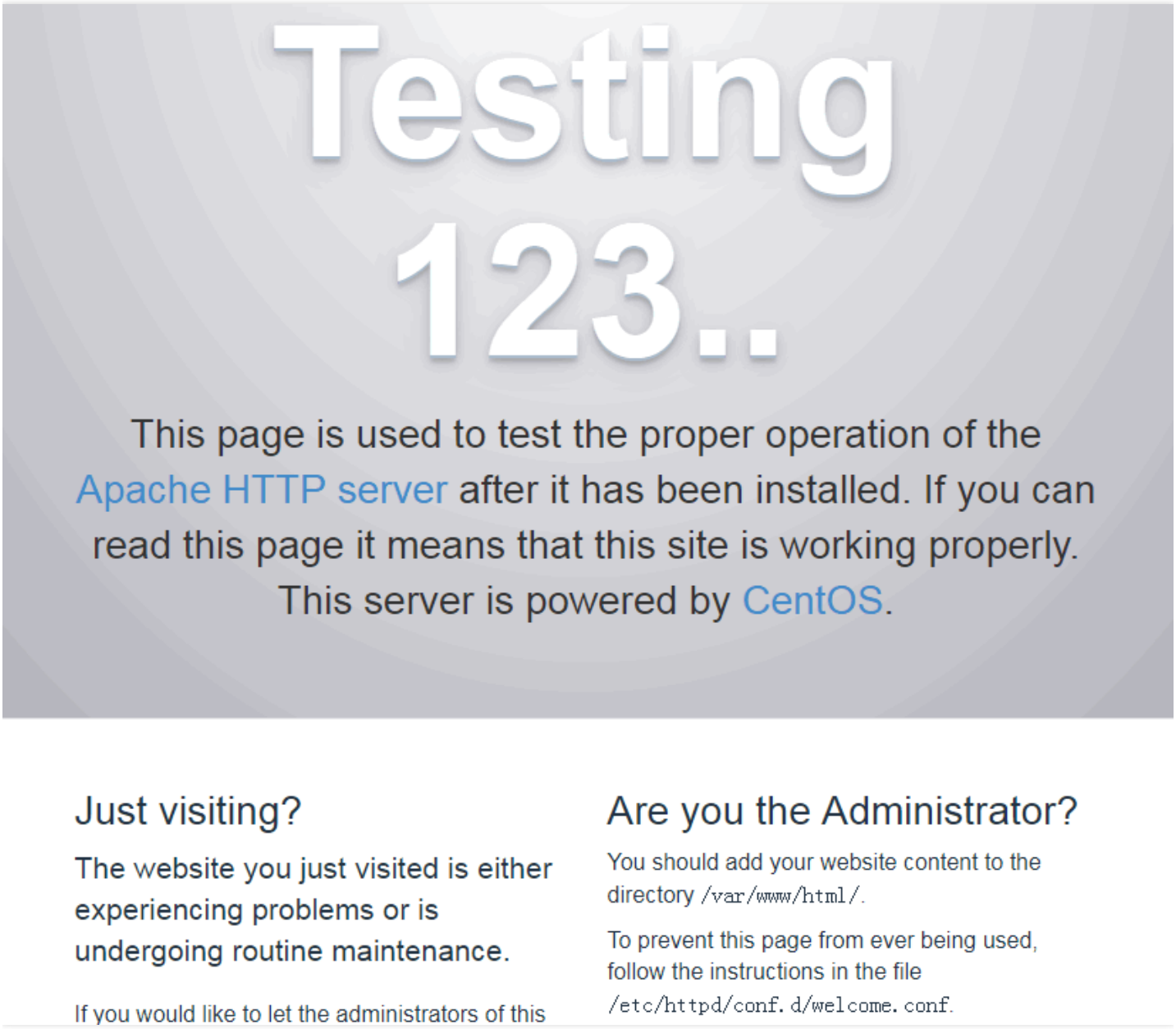
2. 启动 Apache 服务。

```
service httpd start
```

3. 测试 Apache 。

> **注意：**
> 此步骤需要您的云服务器在安全组中配置来源为 **all**，端口协议为 **TCP:80** 的入站规则。关于安全组的配置方法请参考 安全组。

在您本地的浏览器中输入 `http://115.xxx.xxx.xxx/`（其中 `115.xxx.xxx.xxx` 为您的云服务器公网 IP 地址），出现下列画面表示 Apache 启动成功。



安装 Apache 的 mod_wsgi 拓展作为 Django 的应用容器

1. 安装 httpd-devel。

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

2. 安装 mod_wsgi。

```
yum install -y mod_wsgi
```

3. 在httpd.conf文件中添加以下内容，httpd.conf 的路径为 `/etc/httpd/conf/httpd.conf` 。

```
LoadModule wsgi_module modules/mod_wsgi.so
```

创建项目测试 Django 环境

1. 在 /usr/local 下创建测试项目，运行 `django-admin.py startproject projectname` 来创建一个项目，其中 projectname 为项目名。

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

2. 在项目根目录中新建文件 django.wsgi 作为 Apache 支持。

```
cd /usr/local/testProject
vim django.wsgi
```

3. 在 django.wsgi 中输入以下内容。

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

4. 在 Apache 中添加支持 。

```
WSGIScriptAlias /python "/usr/local/testProject/django.wsgi"
```

5. 创建视图，在项目目录下创建 view.py 文件作为访问入口，内容如下。

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

6. 配置 URL，配置项目目录下的 urls.py 文件，删除原来的内容，添加内容如下。

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

7. 重启 Apache 服务。

```
service httpd restart
```

8. 在您本地的浏览器中输入 `http://115.xxx.xxx.xxx/python/hello` （其中 `115.xxx.xxx.xxx` 为您的云服务器公网 IP 地址），页面出现 “Hello world !” 表示项目环境搭建成功。

在 Django 中配置云数据库 CDB（可选）

1. 配置项目目录下的 settings.py 文件。

```
DATABASES = {
'default': {
'ENGINE': 'django.db.backends.mysql',
'NAME': 'mysql',
'USER': 'root', # CDB 账户名
'PASSWORD': '123456', # CDB 账户密码
'HOST': '0.0.0.0', # CDB 内网地址
'PORT': '3306', # CDB 端口
}
}
```

2. 配置完成后可以使用以下命令测试数据库连接。

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

3. 测试通过即可进行数据库操作，有关更多的数据库操作请参考 模型和数据库。

构建 LAMP 堆栈 Web 应用程序

最近更新时间：2018-07-27 17:17:38

LAMP 指 Linux+Apache+Mysql/MariaDB+Perl/PHP/Python，是一组常用来搭建动态网站或者服务器的开源软件。个程序都是独立的，但是因为常被一起使用，相互间的兼容性越来越高，共同组成了一个强大的 Web 应用程序平台。
本教程将指导您完成以下过程：启动一个腾讯云数据库 CDB 实例，通过腾讯云 CVM 配置一个 LAMP 应用程序，以连接腾讯云数据库 CDB 实例的高可用性环境。
运行腾讯云数据库实例会将数据库与环境的生命周期分离，这让您可以从多个服务器连接到同一个数据库，并且简化数据库的运维，让您无需再关心数据库的安装、部署、版本更新及故障处理等问题。

> **注意：**
> 本教程中使用到的云数据库实例和云服务器实例处于同一地域下，如果您的云数据库实例和云服务器实例不处于同一地域下，请参考 外网访问。

初始化云数据库实例

云数据库的购买和初始化请参考 购买与续费 和 初始化 MySQL 数据库。

登录到云服务器实例

云服务器的购买和访问请参考 快速入门 Linux 云服务器，本教程中使用的是 CentOS 系统。

安装 MySQL 客户端

1. 在云服务器实例中使用 `yum` 安装 MySQL 客户端：

```
yum install mysql -y
```

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

Dependencies Resolved

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

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

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

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

Complete!
```

2. 安装完成后，连接到腾讯云数据库实例：

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

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

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

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

MySQL [(none)]>
```

其中，hostname 为数据库实例的内网 IP 地址，username 为您的数据库用户名。

3. 连接成功后，即可退出数据库，进行下一步操作。

```
quit;
```

安装 Apache 服务

1. 在云服务器实例中使用 `yum` 安装 Apache：

```
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!
```

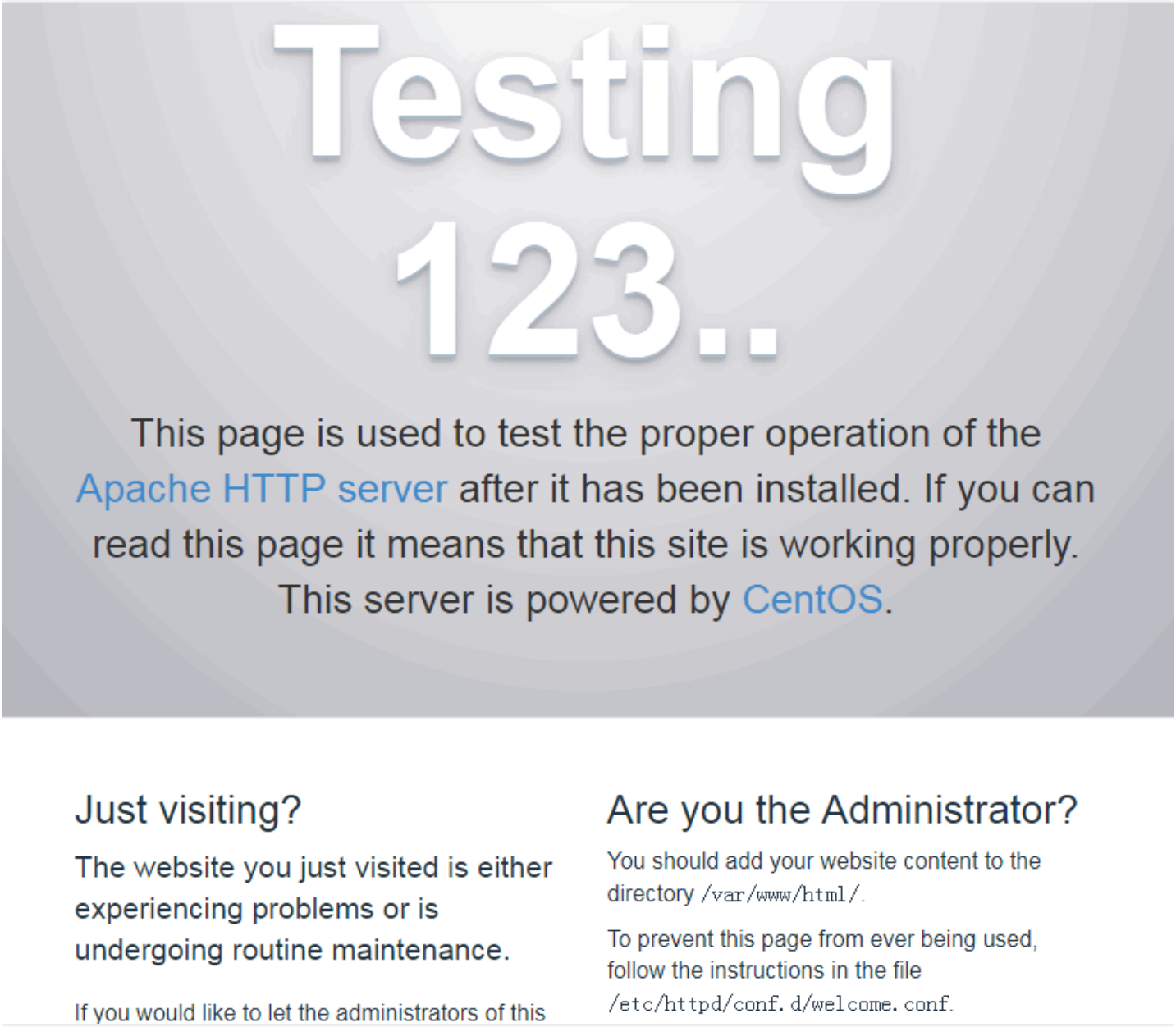
2. 启动 Apache 服务：

```
service httpd start
```

3. 测试 Apache ：

> **注意：**
> 此步骤需要您的云服务器在安全组中配置来源为 **all**，端口协议为 **TCP:80** 的入站规则。关于安全组的配置方法请参考 安全组。

在您本地的浏览器中输入 `http://115.xxx.xxx.xxx/`（其中 `115.xxx.xxx.xxx` 为您的云服务器公网 IP 地址），出现下列画面表示 Apache 启动成功。



安装 PHP

1. 在云服务器实例中使用 `yum` 安装 PHP：

```
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
```

创建项目测试 LAMP 环境

1. 在云服务器 `/var/www/html` 目录下创建一个 info.php 文件，示例代码参考如下：

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

2. 重启 Apache 服务：

```
service httpd restart
```

3. 在您本地的浏览器中输入 `http://0.0.0.0/info.php` ，其中 `0.0.0.0` 为您的云服务器公网 IP 地址，出现下列画面表示 LAMP 服务部署成功。

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

构建 Drupal 网站

最近更新时间：2018-07-27 17:15:19

Drupal 是使用 PHP 语言编写的开源内容管理框架（Content Management Framework），它由内容管理系统（Content Management System）和 PHP 开发框架（Framework）共同构成。Drupal 可用于构造提供多种功能和服务的动态网站，能支持从个人博客到大型社区等各种不同应用的网站项目。
您可以通过本教程了解如何在腾讯云服务器 CVM 上搭建 Drupal 电子商务网站。使用的软件环境为：centos7.2 | Drupal7.56 | PHP5.4.16。

登录到云服务器实例

云服务器的购买和访问请参考 快速入门 Linux 云服务器。

安装 MariaDB 服务

1. CentOS7 以上版本默认支持 MariaDB 数据库，因此我们将使用 MariaDB 数据库。在云服务器实例中使用 `yum` 安装 MariaDB 服务。

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

2. 启动 MariaDB 服务。

```
systemctl start mariadb
```

3. 为 Drupal 创建数据库。（本项目中使用 drupal 作为数据库名）

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

其中，drupal 为 Drupal 服务使用的数据库名。

4. 为数据库创建用户。

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

为用户授权，授权成功后退出数据库。

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

其中，username 为 Drupal 服务使用的数据库用户名，password 为 Drupal 服务使用的数据库密码。

安装 Apache 服务

1. 在云服务器实例中使用 `yum` 安装 Apache。

```
yum install httpd -y
```

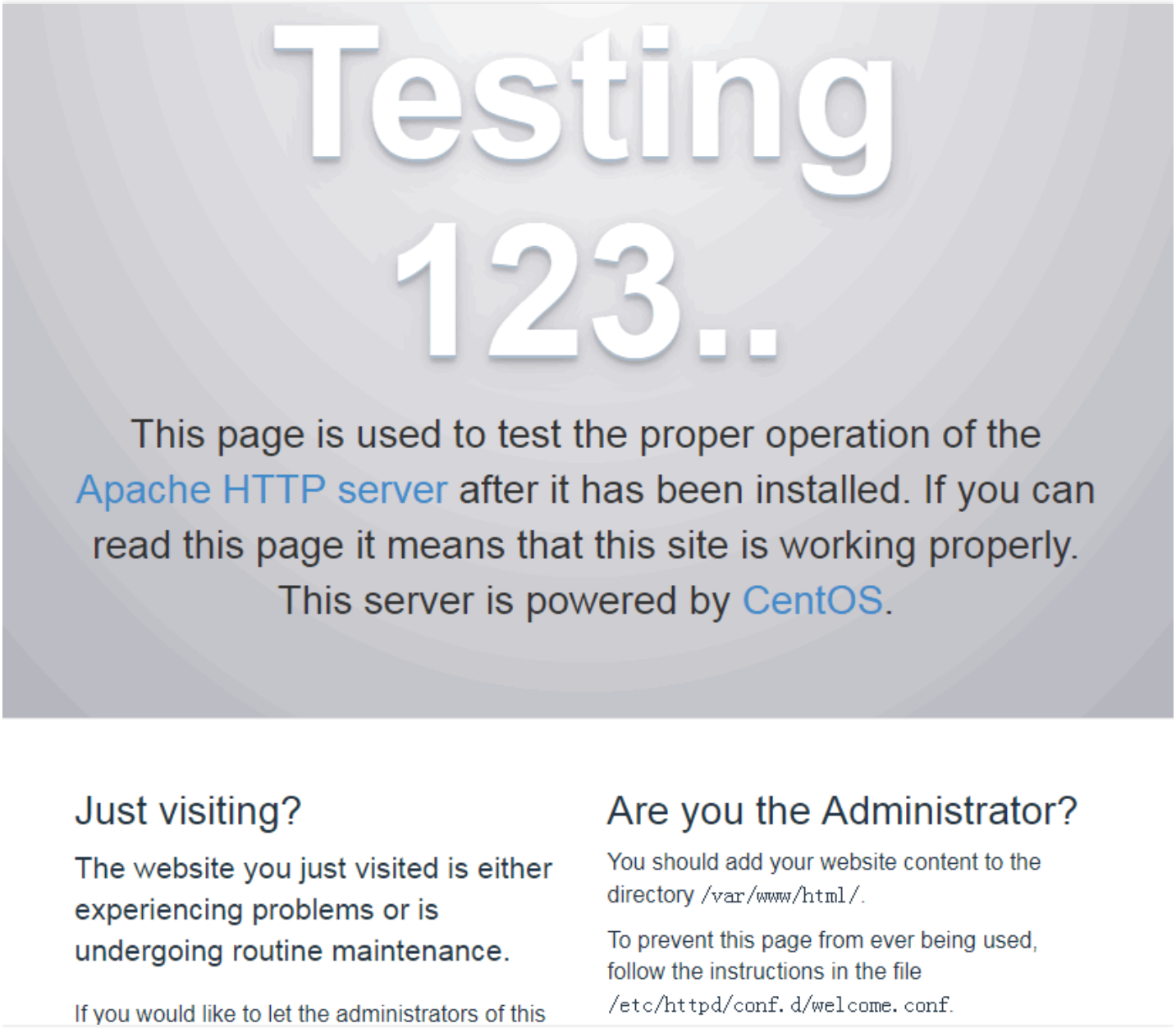
2. 启动 Apache 服务。

```
service httpd start
```

3. 测试 Apache 。

> **注意：**
> 此步骤需要您的云服务器在安全组中配置来源为 **all**，端口协议为 **TCP:80** 的入站规则。关于安全组的配置方法请参考 安全组。

在您本地的浏览器中输入 `http://115.xxx.xxx.xxx/` （其中 `115.xxx.xxx.xxx` 为您的云服务器公网 IP 地址），出现下列画面表示 Apache 启动成功。



安装 PHP

1. 在云服务器实例中使用 `yum` 安装 PHP 及其扩展。

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

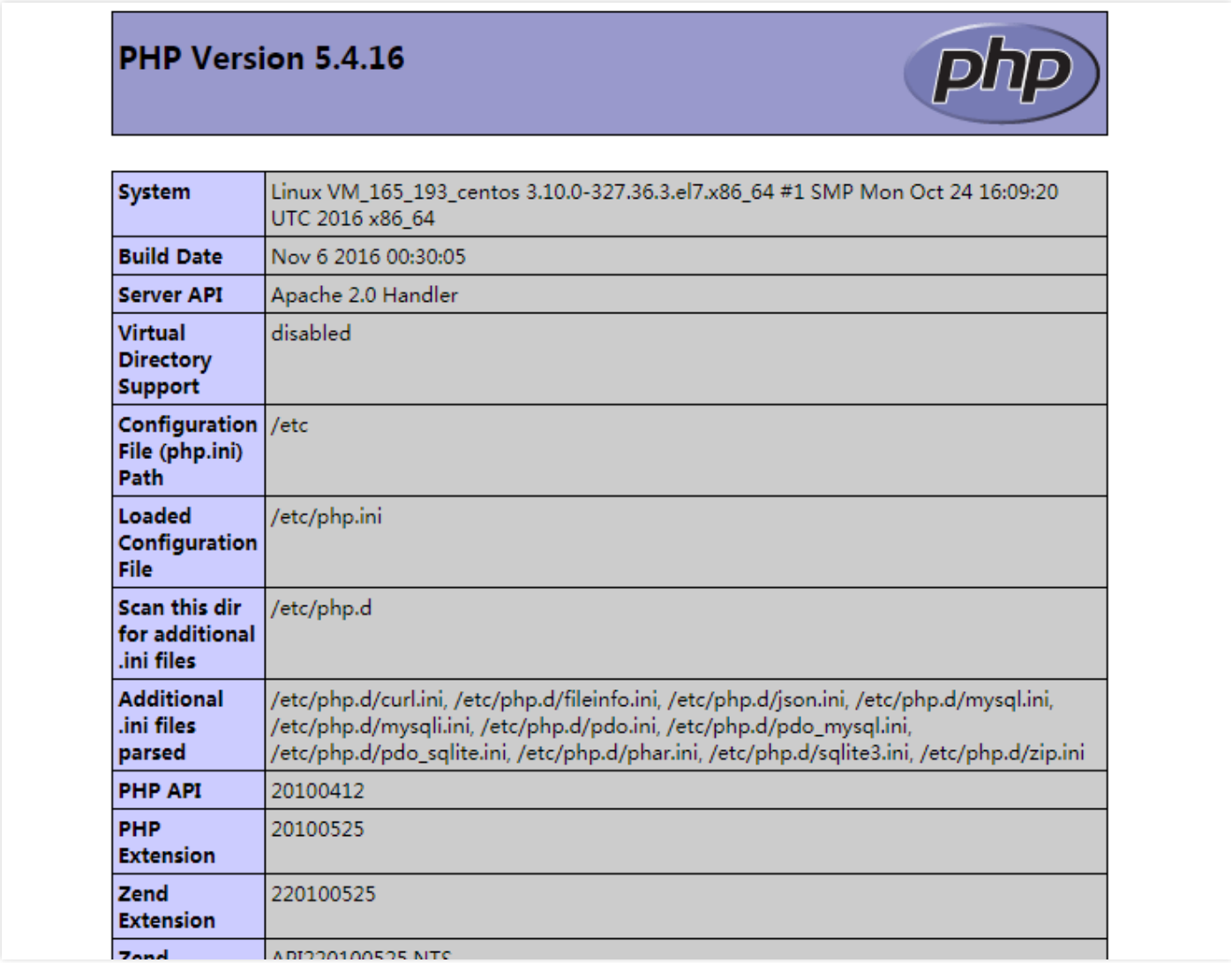
2. 我们在云服务器 `/var/www/html` 目录下创建一个 info.php 文件来检查 PHP 是否安装成功，示例代码参考如下。

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

3. 重启 Apache 服务。

```
service httpd restart
```

4. 在您本地的浏览器中输入 `http://115.xxx.xxx.xxx/info.php`（其中 `115.xxx.xxx.xxx` 为您的云服务器公网 IP 地址），出现下列画面表示 PHP 安装成功。



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

安装 Drupal 服务

1. 下载 Drupal 安装包。

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

2. 解压到网站根目录。

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

3. 下载中文翻译包。

```
cd /var/www/html/
wget -P profiles/standard/translations http://ftp.drupal.org/files/translations/7.x/drupal/drupal-7.56.zh-hans.po
```

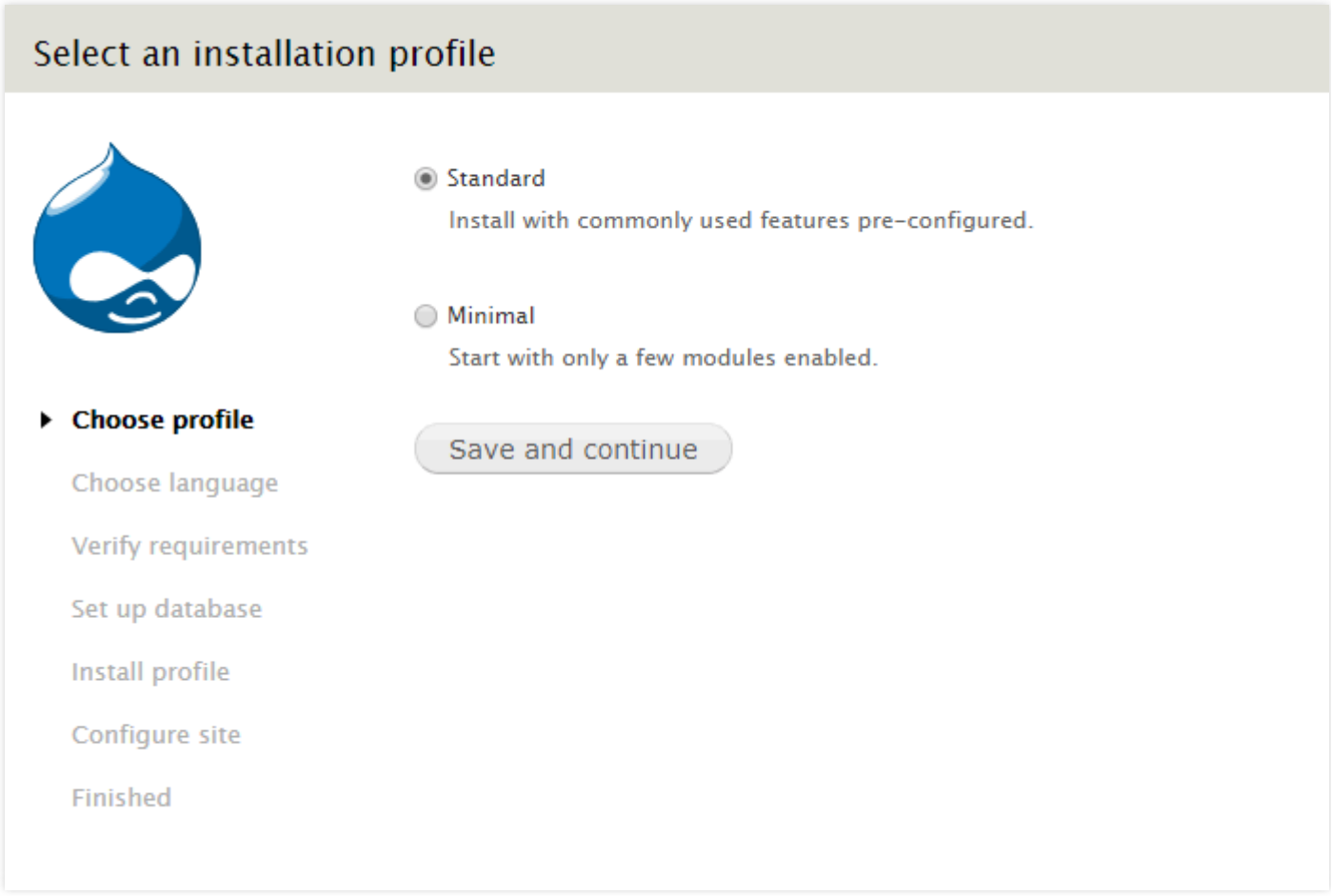
4. 修改 `sites` 目录属主属组。

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

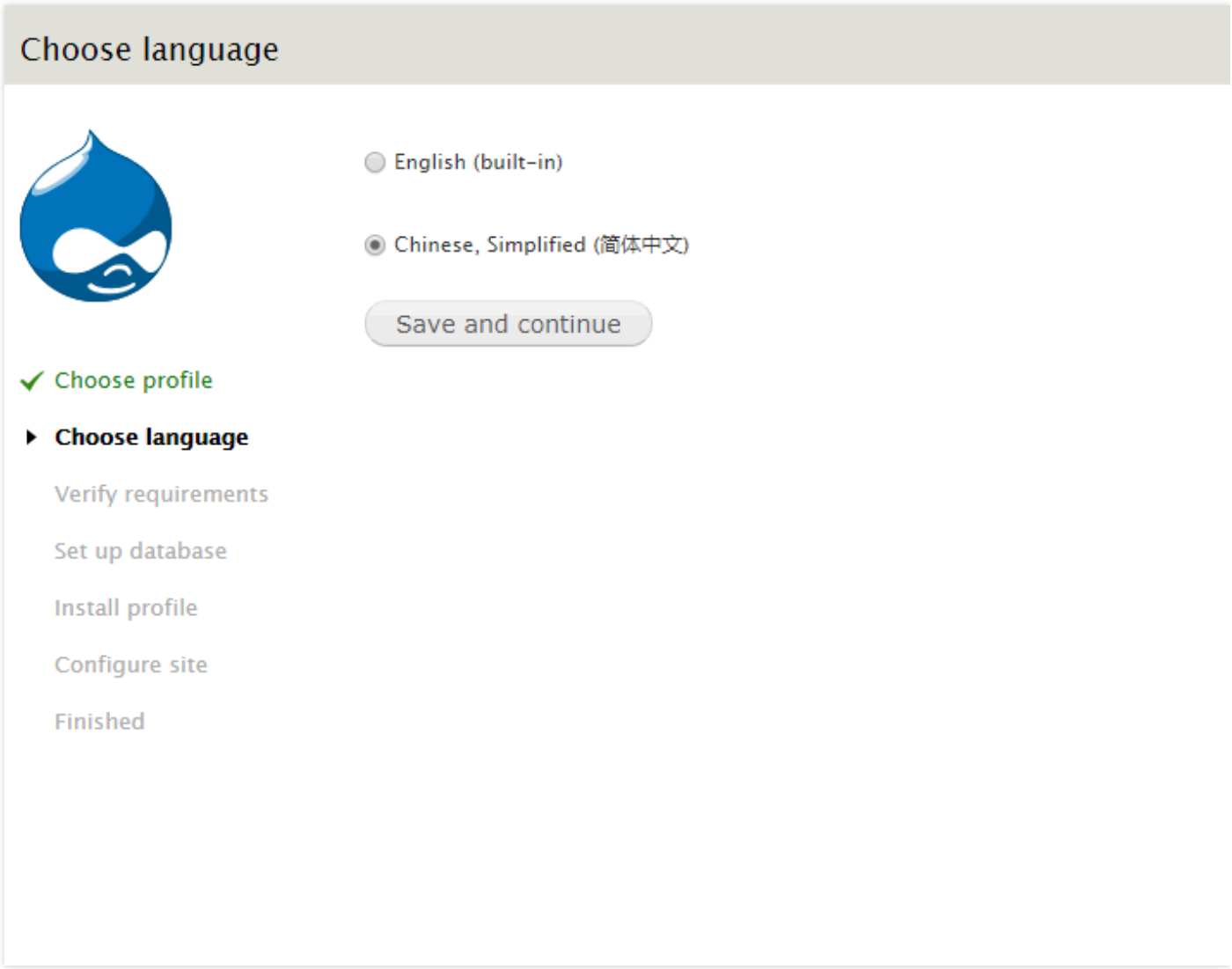
5. 重启 Apache 服务。

```
service httpd restart
```

6. 在您本地的浏览器中输入 `http://115.xxx.xxx.xxx/`（其中 `115.xxx.xxx.xxx` 为您的云服务器公网 IP 地址）,进入 Drupal 安装界面。选择安装版本，单击【Save and continue】。



7. 选择安装语言，单击【Save and continue】。



8. 设置数据库，输入您在 **安装 mariadb 服务** 中配置的数据库信息。

9. 输入站点信息。


选择安装配置文件
选择语言
检查安装需求
设置数据库
按设置安装
安装翻译
设置网站
完成翻译
已完成
站点信息
站点名称 *
站点邮箱 *
自动生成的电子邮件（如注册信息）将从此地址发送。使用您的站点域名作为邮件服务器有助于避免这些邮件被视作垃圾邮件。
站点维护帐号
用户名 *
允许空格，不允许\\\"-\\\"、“-”、“_”以外的其他符号。
电子邮件地址 *
密码 *
密码强度：
确认密码 *
服务器设置
默认国家
- 无 -
为站点选择它的国家。
默认时区
UTC：星期四, 八月 17, 2017 - 08:02 +0000
网站日期会以默认所选择的时区来呈现。
更新通知

0. 完成 Drupal 的安装。

1. 后续可以访问 `http://115.xxx.xxx.xxx/`（其中 `115.xxx.xxx.xxx` 为您的云服务器公网 IP 地址）对网站进行个性化设置。


信息面板 内容 结构 外观 人员 模块 配置 报告 帮助
你好 gart 退出
添加内容 查找内容
编辑快捷键
我的帐户
首页
欢迎光临
首页无内容！
添加新内容
导航
添加内容

高IO版性能测试报告

最近更新时间：2017-12-21 11:43:42

测试工具

数据库基准性能测试为 sysbench 0.5。
工具修改说明：
对 sysbench 自带的 otlp 脚本做了修改，读写比例修改为 1：1，并通过执行测试命令参数 oltp_point_selects 和 oltp_index_updates 来控制读写比例，本文测试用例的均采用 4 个 select 点，1 个 update 点，读写比例保持 4：1。

测试环境

类型	说明
实例物理机器	高 IO 版-单机器最高可支撑 488 GB 内存 6 T 硬盘数据库
实例规格	当前售卖主流配置规格（详见下文测试用例）
客户端配置	4 核 8 GB 内存
客户端数量	1~6 个(配置的提升，客户端数量也需要相应提升)
网络环境	万兆网络机房，网络延时 < 0.05 ms
环境负载	安装 mysql 机器负载 > 70% (针对非独占实例)

- 客户端规格说明：机器采用了较高配置的客户机器，保证单客户端可以压测出数据库实例的性能，如果客户端配置规格较小，建议采用多个客户并行压测实例来求取数据总和。
- 网络延时说明：测试环境保证客户端机器与数据库实例在同一可用区，测试结果不受网络环境影响。

测试方法

1. 测试库表结构

```
CREATE TABLE `sbtest1` (
`id` int(10) unsigned NOT NULL AUTO_INCREMENT,
`k` int(10) unsigned NOT NULL DEFAULT '0',
```

```
`c` char(120) NOT NULL DEFAULT '',
`pad` char(60) NOT NULL DEFAULT '',
PRIMARY KEY (`id`), KEY `k_1` (`k`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ;
```

2. 测试数据行格式

```
id: 1
k: 20106885
c: 08566691963-88624912351-16662227201-46648573979-64646226163-77505759394-7547
0094713-41097360717-15161106334-50535565977
pad: 63188288836-92351140030-06390587585-66802097351-4928296184
```

3. 数据准备

```
/root//sysbench/sysbench --mysql-host=xxxx --mysql-port=xxxx --mysql-user=xxx --mysql-password
=xxx --mysql-db=test --mysql-table-engine=innodb --test=tests/db/oltp.lua --oltp_tables_count=20
--oltp-table-size=10000000 --rand-init=on prepare
```

数据准备参数说明：

- `--test=tests/db/oltp.lua` ，表示调用 tests/db/oltp.lua 脚本进行 oltp 模式测试。
- `--oltp_tables_count=20` ，表示用于测试的表数量为 20 张。
- `--oltp-table-size=10000000` ，表示每个测试表填充数据行数为 1000 W 行。
- `--rand-init=on` ，表示每个测试表都是用随机数据来填充的。

4. 性能压测命令

```
/root//sysbench/sysbench --mysql-host=xxxx --mysql-port=xxx --mysql-user=xxx --mysql-password=
xxx --mysql-db=test --test=/root/sysbench_for_z3/sysbench/tests/db/oltp.lua --oltp_tables_count=xx
--oltp-table-size=xxxx --num-threads=xxx --oltp-read-only=off --rand-type=special --max-time=600
--max-requests=0 --percentile=99 --oltp-point-selects=4 run
```

性能压测参数说明：

- `--test=/root/sysbench_for_z3/sysbench/tests/db/oltp.lua` ，表示调用 /root/sysbench_for_z3/sysbench/tests/db/oltp.lua 脚本进行 oltp 模式测试。。
- `--oltp_tables_count=20` ，表示本次用于测试的表数量为 20 张。
- `--oltp-table-size=10000000` ，表示本次测试使用的表行数均为 1000 W 行。
- `--num-threads=128` ，表示本次测试的客户端连接并发数为 128。
- `--oltp-read-only=off` ，off 表示测试关闭只读测试模型，采用读写混合模型。
- `--rand-type=special` ，表示随机模型为特定的。

- `--max-time=1800` ，表示本次测试的执行时间。
- `--max-requests=0` ，0 表示不限制总请求数，而是按 max-time 来测试。
- `--percentile=99` ，表示设定采样比例，默认是 95%，即丢弃 1% 的长请求，在剩余的 99% 里取最大值。
- `--oltp-point-selects=4` ，表示 oltp 脚本中 sql 测试命令，select 操作次数为 4，默认值为 1。

5. 场景模型

本文用例均使用场景脚本 our_oltp.lua，修改为 4 个 select 点查询，1 个 update （索引列），读写比为 4：1。针对最大配置类型，对数据场景增加了参数调优模型，测试结果见下文 测试结果。

测试参数

内存	存储空间	表数量	表行数	数据集大小	并发数	执行时间(m)
4 GB	200 GB	8	4000 W	76 GB	128	30
8 GB	200 GB	15	4000 W	142 GB	128	30
16 GB	400 GB	25	4000 W	238 GB	128	30
32 GB	700 GB	25	4000 W	238 GB	128	30
64 GB	1 T	40	4000 W	378 GB	256	30
96 GB	1.5 T	40	4000 W	378 GB	128	30
128 GB	2 T	40	4000 W	378 GB	128	30
244 GB	3 T	60	4000 W	567 GB	128	30
488 GB	6 T	60	4000 W	567 GB	128	30
488 GB(调优)	6 T	60	1000 W	140 GB	128	30

测试结果

内存	存储空间	数据集	客户端数	单客户端并发数	QPS	TPS
4 GB	200 GB	76 GB	1	128	4082	816
8 GB	200 GB	142 GB	1	128	6551	1310

内存	存储空间	数据集	客户端数	单客户端并发数	QPS	TPS
16 GB	400 GB	238 GB	1	128	11098	2219
32 GB	700 GB	238 GB	2	128	20484	3768
64 GB	1 T	378 GB	2	128	36395	7279
96 GB	1.5 T	378 GB	3	128	56464	11292
128 GB	2 T	378 GB	3	128	81752	16350
244 GB	3 T	567 GB	4	128	98528	19705
488 GB	6 T	567 GB	6	128	142246	28449
488 GB(调优)	6 T	140 GB	6	128	245509	46304

MySQL主实例参数修改的影响

最近更新时间：2018-09-13 09:33:11

对于云数据库 MySQL 实例，您可以通过 控制台 修改主实例的参数。其中对于某些重要参数而言，使用不恰当的修改方式会导致灾备实例异常或数据不一致，本文将介绍如下重要参数修改后的影响。

lower_case_table_names

默认值：0
作用：创建数据库及表时，存储与查询时是否大小写敏感。该参数可以设置的值为 0、1，默认的参数值为 0，表示创建数据库及表时，存储与查询均区分大小写，反之则不做区分。
影响：主实例修改参数后，无法同步修改灾备实例的参数，当主实例大小写敏感而灾备实例大小写不敏感时，比如主实例创建两张表，表名分别为 Test、TEst 时，当灾备实例在应用对应日志时，会导致数据同步状态异常，而错误原因为TEst 表名已存在。

auto_increment_increment

默认值：1
作用：用于自增列 AUTO_INCREMENT 的增量值，该参数可以设置的范围为 1-65535，默认值为 1。
影响：主实例修改参数后，无法同步修改灾备实例的参数，当主实例修改自增列的增量值，而灾备实例未同步更改，会导致主、备实例的数据不一致。

auto_increment_offset

默认值：1
作用：用于自增列 AUTO_INCREMENT 的起始值（偏移量），该参数可以设置的范围为 1-65535，默认值为 1。
影响：主实例修改参数后，无法同步修改灾备实例的参数，当主实例修改自增列的起始值，而灾备实例未同步更改，会导致主、备实例的数据不一致。

sql_mode

默认值：NO_ENGINE_SUBSTITUTION
作用：MySQL 可以运行在不同 sql mode 模式，sql mode 模式定义了 mysql 应该支持的 sql 语法，数据校验等。该参数 5.6 版本的默认参数值为 NO_ENGINE_SUBSTITUTION，表示使用的存储引擎被禁用或未编译则抛出错误；5.7版本的默认参数值为
`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`，
其中：

- `ONLY_FULL_GROUP_BY` 表示在 GROUP BY 聚合操作时，如果在 SELECT 中的列、HAVING 或者 ORDER BY 子句的列，必须是 GROUP BY 中出现或者依赖于 GROUP BY 列的函数列；
- `STRICT_TRANS_TABLES` 为启用严格模式；

- NO_ZERO_IN_DATE 是否允许日期中的月份和日包含 0，且受是否开启严格模式的影响；
- NO_ZERO_DATE 数据库不允许插入零日期，且受是否开启严格模式的影响；
- ERROR_FOR_DIVISION_BY_ZERO 在严格模式下，INSERT 或 UPDATE 过程中，如果数据被零除，则产生错误而非警告，而非严格模式下，数据被零除时 MySQL 返回 NULL；
- NO_AUTO_CREATE_USER 禁止 GRANT 创建密码为空的用户；
- NO_ENGINE_SUBSTITUTION 使用的存储引擎被禁用或者未编译则抛出错误；

影响：主实例修改参数后，无法同步修改灾备实例的参数，当主实例修改了 sql mode 模式，而灾备实例未同步更改，如主实例 sql mode 模式限制小于灾备实例 sql mode 模式的限制，可能会出现在主实例执行成功的 SQL 同步至灾备实例时出现报错，进而导致主、备实例数据不一致。

通过Python语言使用MySQL API
实例购买

最近更新时间：2018-08-29 09:44:49

API	描述
CreateDBInstance	创建云数据库实例（包年包月）
CreateDBInstanceHour	创建云数据库实例（按量计费）
DescribeDBInstances	查询实例列表
DescribeDBPrice	查询数据库价格
DescribeDBZoneConfig	获取云数据库可售卖规格
InitDBInstances	初始化新实例

1、CreateDBInstance 创建云数据库实例（包年包月）

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

# 引入云API入口模块
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKExcept
ion
from tencentcloud.cdb.v20170320 import cdb_client, models


'''购买主实例'''
def CreateDBInstancedemomaster():
try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-beijing")
```

```
#实例化一个请求对象: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"


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.CreateDBInstance(req)
# 输出json格式的字符串回包
print(resp.to_json_string())


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




'''购买只读实例'''
def CreateDBInstancedemoro():
try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-beijing")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.CreateDBInstanceRequest()
req.Memory = 2000
req.Volume = 200
req.Period = 1
req.GoodsNum =1
```

```
req.Zone = "ap-beijing-1"
req.Port = 3306
req.InstanceRole = "ro"
req.EngineVersion = "5.6"
req.Password = "CDB@Qcloud"
req.ProtectMode = 0
req.DeployMode =1
req.GoodsNum =2
req.SlaveZone = "ap-beijing-1"
req.ParamList = [{"name":"max_connections","value":"1000"},{"name":"lower_case_table_names","valu
e":"1"}]
req.BackupZone = "0"
req.AutoRenewFlag =0
req.MasterInstanceId ="cdb-bgr97hu0"
req.RoGroup = {"RoGroupMode":"allinone","RoGroupName":"roweek"}
req.InstanceName = "tencentcdbRO"


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.CreateDBInstance(req)
# 输出json格式的字符串回包
print(resp.to_json_string())


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


'''购买灾备实例'''
def CreateDBInstancedemodr():
try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.CreateDBInstanceRequest()

req.Memory = 4000
req.Volume = 200
req.Period = 1
```

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


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


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.CreateDBInstance(req)
# 输出json格式的字符串回包
print(resp.to_json_string())


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

#CreateDBInstancedemodr()
```

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

2、CreateDBInstanceHour 创建云数据库实例（按量计费）

```
'''小时计费要冻结金额，需要账号有钱，如果账号余额为0，则不能购买'''
#!/usr/bin/python
# -*- coding: utf-8 -*-

# 引入云API入口模块
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-beijing")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.CreateDBInstanceHourRequest()
req.EngineVersion = "5.6"
req.Zone = "ap-beijing-3"
req.ProjectId = 0
req.GoodsNum = 1
req.Memory = 1000
req.Volume = 50
req.InstanceRole = "master"
req.Port =3311
req.Password = "CDB@Qcloud"
req.ParamList = [{"name":"max_connections","value":"1000"},{"name":"lower_case_table_names","value":"1"}]
req.ProtectMode = 1
req.SlaveZone = "ap-beijing-3"
req.InstanceName = "oneday1"
req.AutoRenewFlag = 0
```

```
# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.CreateDBInstanceHour(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
msg = traceback.format_exc() # 方式1
print (msg)
```

3、DescribeDBInstances 查询实例列表

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

# 引入云API入口模块
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKExcept
ion
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.DescribeDBInstancesRequest()
req.EngineVersions = ["5.6"]
req.OrderBy = "instanceId"
req.InstanceIds = ["cdb-1j8lumf6"]


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.DescribeDBInstances(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
```

```
msg = traceback.format_exc() # 方式1
print (msg)
```

4、DescribeDBPrice 查询数据库价格

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

# 引入云API入口模块
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKExcept
ion
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-guangzhou")

#实例化一个请求对象: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


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.DescribeDBPrice(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
print(err)
```

5、DescribeDBZoneConfig 获取云数据库可售卖规格

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

```
# 引入云API入口模块

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

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.DescribeDBZoneConfigRequest()
#req.InstanceId = "cdb-j0edpju5"


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.DescribeDBZoneConfig(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
print(err)
```

6、InitDBInstances 初始化新实例

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

# 引入云API入口模块

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

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
```

```
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.InitDBInstancesRequest()
req.InstanceIds = ["cdb-c752yqcn"]
req.NewPassword = "CDB@Qcloud"

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


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.InitDBInstances(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
print(err)
```

实例管理

最近更新时间：2018-08-29 09:47:57

API	描述
ModifyInstanceParam	修改实例参数
CloseWanService	关闭实例外网访问
OpenWanService	开通实例外网访问
RestartDBInstances	重启实例
OpenDBInstanceGTID	开启实例的GTID
ModifyDBInstanceName	修改云数据库实例名
ModifyDBInstanceProject	修改云数据库实例的所属项目
ModifyDBInstanceVipVport	修改云数据库实例的IP和端口号
DescribeDBInstanceCharset	查询云数据库实例的字符集
DescribeDBInstanceConfig	查询云数据库实例的配置信息
DescribeDBInstanceGTID	查询云数据实例的GTID是否开通
DescribeDBInstanceRebootTime	查询云数据库实例的预期重启时间

1、ModifyInstanceParam 修改实例参数

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

# 引入云API入口模块
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKExcept
ion
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
```

```
bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象
req = models.ModifyInstanceParamRequest()
req.InstanceIds = ["cdb-1y6g3zj8","cdb-7ghaiocc"]
req.ParamList = [{"name":"max_connections","currentValue":"100"},{"name":"character_set_server","cur
rentValue":"utf8"},{"name":"lower_case_table_names","currentValue":"1"}]
#req.ParamList = [{"name":"max_connections","currentValue":"100"}]
#param = models.Parameter()
#param.Name = "max_connections"
#param.CurrentValue = "1000"
#req.ParamList = [param]


print req
# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.ModifyInstanceParam(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
msg = traceback.format_exc() # 方式1
print (msg)
```

2、CloseWanService 关闭实例外网访问

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

# 引入云API入口模块

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

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")
```

```
#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.CloseWanServiceRequest()
req.InstanceId = "cdb-1y6g3zj8"

# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.CloseWanService(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
print(err)
```

3、OpenWanService 开通实例外网访问

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

# 引入云API入口模块

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

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.OpenWanServiceRequest()
req.InstanceId = "cdb-1y6g3zj8"

# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.OpenWanService(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
```

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

4、RestartDBInstances 重启实例

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

# 引入云API入口模块

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

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.RestartDBInstancesRequest()
req.InstanceIds = ["cdb-7ghaiocc"]

# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.RestartDBInstances(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
print(err)
```

5、OpenDBInstanceGTID 开启实例的GTID

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

# 引入云API入口模块

from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKExcept
```

```
ion
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.OpenDBInstanceGTIDRequest()
req.InstanceId = "cdb-7ghaiocc"


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.OpenDBInstanceGTID(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
print(err)
```

6、ModifyDBInstanceName 修改云数据库实例名

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

# 引入云API入口模块
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKExcept
ion
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-beijing")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
```

```
req = models.ModifyDBInstanceNameRequest()
req.InstanceId = "cdb-cukm86n2"
req.InstanceName = "1s中文"

# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.ModifyDBInstanceName(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
print(err)
```

7、ModifyDBInstanceProject 修改云数据库实例的所属项目

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

# 引入云API入口模块
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKExcept
ion
from tencentcloud.cdb.v20170320 import cdb_client, models


def DescribeDBInstancesList():
try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.ModifyDBInstanceProjectRequest()
req.InstanceIds = ["cdb-7ghaiocc"]
req.NewProjectId =1


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.ModifyDBInstanceProject(req)
```

```
# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
msg = traceback.format_exc() # 方式1
print (msg)



DescribeDBInstancesList()
```

8、ModifyDBInstanceVipVport 修改云数据库实例的 IP 和端口号

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

# 引入云API入口模块
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKExcept
ion
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.ModifyDBInstanceVipVportRequest()
req.InstanceId = "cdb-7ghaiocc"
req.DstIp = "10.0.0.13"
req.DstPort =1025
req.UniqVpcId = 1111

# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.ModifyDBInstanceVipVport(req)

# 输出json格式的字符串回包
```

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

9、DescribeDBInstanceCharset 查询云数据库实例的字符集

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

# 引入云API入口模块

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

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.DescribeDBInstanceCharsetRequest()
req.InstanceId = "cdb-1y6g3zj8"


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.DescribeDBInstanceCharset(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
print(err)
```

10、DescribeDBInstanceConfig 查询云数据库实例的配置信息

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

# 引入云API入口模块
```

```
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKExcept
ion
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.DescribeDBInstanceConfigRequest()
req.InstanceId = "cdb-1y6g3zj8"


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.DescribeDBInstanceConfig(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
print(err)
```

11、DescribeDBInstanceGTID 查询云数据实例的 GTID 是否开通

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

# 引入云API入口模块

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

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")
```

```
#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.DescribeDBInstanceGTIDRequest()
req.InstanceId = "cdb-1y6g3zj8"


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.DescribeDBInstanceGTID(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
print(err)
```

12、DescribeDBInstanceRebootTime 查询云数据库实例的预期重启时间

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

# 引入云API入口模块

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

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("AKIDrY4UwCdhewLbhdG7f2DmDT4eFftKtWIz", "eGl4KUSEsmaFiyEKFID0
bzFKNFvPh2EE")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.DescribeDBInstanceRebootTimeRequest()
req.InstanceIds = ["cdb-1y6g3zj8"]


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.DescribeDBInstanceRebootTime(req)
```

```
# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
print(err)
```

备份任务

最近更新时间：2018-08-30 10:13:54

API	描述
CreateDBInstance	创建云数据库实例（包年包月）
CreateDBInstanceHour	创建云数据库实例（按量计费）
DescribeDBInstances	查询实例列表
DescribeDBPrice	查询数据库价格
DescribeDBZoneConfig	获取云数据库可售卖规格
InitDBInstances	初始化新实例

1、CreateDBInstance 创建云数据库实例（包年包月）

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

# 引入云API入口模块
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKExcept
ion
from tencentcloud.cdb.v20170320 import cdb_client, models


'''购买主实例'''
def CreateDBInstancedemomaster():
try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("secretId", "secretKey")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-beijing")

#实例化一个请求对象: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"]


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.CreateDBInstance(req)
# 输出json格式的字符串回包
print(resp.to_json_string())


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



'''购买只读实例'''
def CreateDBInstancedemoro():
try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("secretId", "secretId")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-beijing")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.CreateDBInstanceRequest()
req.Memory = 2000
req.Volume = 200
req.Period = 1
req.GoodsNum =1
req.Zone = "ap-beijing-1"
req.Port = 3306
req.InstanceRole = "ro"
req.EngineVersion = "5.6"
```

```
req.Password = "CDB@Qcloud"
req.ProtectMode = 0
req.DeployMode =1
req.GoodsNum =2
req.SlaveZone = "ap-beijing-1"
req.ParamList = [{"name":"max_connections","value":"1000"},{"name":"lower_case_table_names","valu
e":"1"}]
req.BackupZone = "0"
req.AutoRenewFlag =0
req.MasterInstanceId ="cdb-bgr97hu0"
req.RoGroup = {"RoGroupMode":"allinone","RoGroupName":"roweek"}
req.InstanceName = "tencentcdbRO"


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.CreateDBInstance(req)
# 输出json格式的字符串回包
print(resp.to_json_string())


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


'''购买灾备实例'''
def CreateDBInstancedemodr():
try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("secretId", "secretKey")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.CreateDBInstanceRequest()

req.Memory = 4000
req.Volume = 200
req.Period = 1
req.GoodsNum =1
#req.Zone = "ap-shanghai-2"
req.Port = 3306
req.InstanceRole = "dr"
#req.MasterInstanceId
```

```
req.EngineVersion = "5.6"
req.Password = "CDB@Qcloud"
req.ProtectMode = 0
req.DeployMode =0
#req.SlaveZone = "ap-guangzhou-3"
req.ParamList = [{"name":"max_connections","value":"1000"},{"name":"lower_case_table_names","valu
e":"1"}]
req.BackupZone = "0"
req.AutoRenewFlag =0
#req.RoGroup = {"RoGroupMode":"alone","RoGroupName":"roweek"}
#req.RoGroup = {"RoGroupName":"roweek"}
#param = models.RoGroup()
#param.RoGroupMode = "alone"
#param.RoGroupName = "roweek"
#param.MinRoInGroup = 1
#req.RoGroup = [param]


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


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.CreateDBInstance(req)
# 输出json格式的字符串回包
print(resp.to_json_string())


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

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

2、CreateDBInstanceHour 创建云数据库实例（按量计费）

```
'''小时计费要冻结金额，需要账号有钱，如果账号余额为0，则不能购买'''
#!/usr/bin/python
# -*- coding: utf-8 -*-

# 引入云API入口模块
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKExcept
ion
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("secretId", "secretKey")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-beijing")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.CreateDBInstanceHourRequest()
req.EngineVersion = "5.6"
req.Zone = "ap-beijing-3"
req.ProjectId = 0
req.GoodsNum = 1
req.Memory = 1000
req.Volume = 50
req.InstanceRole = "master"
req.Port =3311
req.Password = "CDB@Qcloud"
req.ParamList = [{"name":"max_connections","value":"1000"},{"name":"lower_case_table_names","valu
e":"1"}]
req.ProtectMode = 1
req.SlaveZone = "ap-beijing-3"
req.InstanceName = "oneday1"
req.AutoRenewFlag = 0


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.CreateDBInstanceHour(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
```

```
msg = traceback.format_exc() # 方式1
print (msg)
```

3、DescribeDBInstances 查询实例列表

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

# 引入云API入口模块
import logging
import traceback
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKExcept
ion
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("secretId", "secretKey")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.DescribeDBInstancesRequest()
req.EngineVersions = ["5.6"]
req.OrderBy = "instanceId"
req.InstanceIds = ["cdb-1j8lumf6"]


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.DescribeDBInstances(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
msg = traceback.format_exc() # 方式1
print (msg)
```

4、DescribeDBPrice 查询数据库价格

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

```
# 引入云API入口模块
from tencentcloud.common import credential
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKExcept
ion
from tencentcloud.cdb.v20170320 import cdb_client, models

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("secretId", "secretKey")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-guangzhou")

#实例化一个请求对象: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


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.DescribeDBPrice(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
print(err)
```

5、DescribeDBZoneConfig 获取云数据库可售卖规格

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

# 引入云API入口模块

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

try:
```

```
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("secretId", "secretKey")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.DescribeDBZoneConfigRequest()
#req.InstanceId = "cdb-j0edpju5"


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.DescribeDBZoneConfig(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
print(err)
```

6、InitDBInstances 初始化新实例

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

# 引入云API入口模块

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

try:
# 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey
cred = credential.Credential("secretId", "secretKey")

#实例化要请求产品(以cdb为例)的client对象
client = cdb_client.CdbClient(cred, "ap-shanghai")

#实例化一个请求对象:req = models.ModifyInstanceParamRequest()
req = models.InitDBInstancesRequest()
req.InstanceIds = ["cdb-c752yqcn"]
req.NewPassword = "CDB@Qcloud"
```

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


# 通过client对象调用想要访问的接口，需要传入请求对象
resp = client.InitDBInstances(req)

# 输出json格式的字符串回包
print(resp.to_json_string())
except TencentCloudSDKException as err:
print(err)
```