云备份

最佳实践

文档版本 01

发布日期 2024-11-21

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 通过自定义脚本实现数据库备份

1.1 通过自定义脚本实现 MySQL 一致性备份

本章节以SuSE 11 SP3操作系统下MySQL 5.5单机版为例，介绍如何通过自定义脚本来冻结、解冻MySQL数据库，以实现对于MySQL数据库的数据库备份。

场景介绍

某企业购买了云服务器，并在云服务器中安装了MySQL 5.5数据库用于存放业务数据。随着数据量的增加，之前的崩溃一致性保护已经满足不了RTO、RPO的要求，决定采用应用一致性备份，减小RTO与RPO。

数据准备

表 1-1 数据准备

准备项	说明	示例
MySQL用户名	连接MySQL数据库时使用的用户名	root
MySQL密码	连接MySQL数据库时使用的密码	Example@123

详细步骤

步骤1 加密MySQL密码，供自定义脚本使用

1. 登录MySQL服务器，输入**cd /home/rdadmin/Agent/bin/** ，进入Agent目录。
2. 执行**/home/rdadmin/Agent/bin/agentcli encpwd**，回显如下：
   ```
   Enter password:
   ```
   输入MySQL密码，并按“Enter”，屏幕上就会打印出加密后的密码，将其复制到剪贴板中。

   说明

   冻结解冻脚本中配置的明文密码长度不超过16位，否则配置后密码会被截断，应用一致性备份会失败。

步骤2 执行**cd /home/rdadmin/Agent/bin/thirdparty/ebk_user** ，进入自定义脚本目录，然后执行**vi mysql_freeze.sh**，打开MySQL示例冻结脚本。

将下图所示的**MYSQL_USER**与**MYSQL_PASSWORD**修改为实际值，其中**MYSQL_PASSWORD**为**步骤1**的屏幕输出。

```
#*********************************************************************************
#Importnata note
#Please change this parameters according to your Mysql system configuration!!!
MYSQL_USER="root"
MYSQL_PASSWORD="00000001000000010000000000000005000000010000000017334dcf36ace871b1
0001000000000000080400000001000000012967889 4e3225391233bac37497d372800000000000000
#*********************************************************************************
```

也可以使用sed命令来直接进行修改：

sed -i 's/^MYSQL_PASSWORD=.*/MYSQL_PASSWORD="*XXX*"/' mysql_freeze.sh mysql_unfreeze.sh ，其中***XXX***为步骤1中打印出的密码。

此操作会同时修改冻结解冻脚本，所以无需再执行**步骤3**。

步骤3 执行**vi mysql_unfreeze.sh**，打开MySQL示例解冻脚本，修改此脚本中的用户名和密码。

mysql_unfreeze.sh与**mysql_freeze.sh**脚本实现了基本的数据库冻结与解冻操作，如果您在冻结、解冻时有其它额外步骤需要执行，可以自行在其中进行修改。详细说明请参见 **通过自定义脚本实现其它Linux应用的一致性备份**。

⚠ 注意

MySQL的冻结是通过FLUSH TABLES WITH READ LOCK指令来实现的，此指令不会触发bin log刷盘操作，如果开启了bin log，且sync_binlog参数不为1，则可能出现保存的备份映像中部分SQL操作未记录到bin log的情况，如果bin log也需要完整保护，请设置sync_binlog=1。

----结束

1.2 通过自定义脚本实现 SAP HANA 一致性备份

本章节以SuSE 11 SP4 for SAP操作系统下HANA 2.0单机版为例，介绍如何通过自定义脚本来冻结、解冻HANA数据库，以实现对HANA数据库的数据库备份。

场景介绍

某企业购买了云服务器，并在上面安装了HANA 2.0单机版数据库，用于存放业务数据，随着数据量的增加，之前的崩溃一致性保护已经满足不了RTO、RPO的要求，决定采用应用一致性备份，减小RTO与RPO。

数据准备

表 1-2 数据准备

准备项	说明	示例
HANA用户名	连接HANA SYSTEMDB 数据库时使用的用户名	system
HANA密码	连接HANA SYSTEMDB 数据库时使用的密码	Example@123
HANA实例编号	连接HANA数据库时使用的实例编号	00
HANA SID	连接HANA数据库时使用的SID	WXJ

详细步骤

步骤1 加密HANA用户密码，供自定义脚本使用

1. 登录HANA服务器，输入**cd /home/rdadmin/Agent/bin/** ，进入Agent目录。
2. 执行**/home/rdadmin/Agent/bin/agentcli encpwd**，回显如下：
   ```
   Enter password:
   ```
   输入HANA用户的密码，并按“Enter”，屏幕上就会打印出加密后的密码，将其复制到剪贴板中。

   说明

   冻结解冻脚本中配置的明文密码长度不超过16位，否则配置后密码会被截断，应用一致性备份会失败。

   执行**cd /home/rdadmin/Agent/bin/thirdparty/ebk_user** ，进入自定义脚本目录，执行**vi hana_freeze.sh** ，打开HANA示例冻结脚本。

步骤2 将下图所示的**HANA_USER HANA_PASSWORD INSTANCE_NUMBER DB_SID**修改为实际值，其中**HANA_PASSWORD** 为步骤1的屏幕输出。

```
#*********************************************************************************
#Importnata note
#Please change this parameters according to your HANA system configuration!!!

HANA_USER="system"
HANA_PASSWORD="00000001000000010000000000000005000000010000000161b3258428fbdf
001000000000000080400000001000000001c9562ef9b7f838e984dc3d1080975be00000000000
INSTANCE_NUMBER="00"
DB_SID="WXJ"

#*********************************************************************************
```

也可以使用sed命令来直接进行修改：

sed -i 's/^HANA_USER=.*/HANA_USER="*XXX*"/' hana_freeze.sh hana_unfreeze.sh ，其中***XXX***为数据库用户名。

sed -i 's/^HANA_PASSWORD=.*/HANA_PASSWORD="*XXX*"/' hana_freeze.sh hana_unfreeze.sh ，其中***XXX***为步骤1中打印出的密码。

sed -i 's/^INSTANCE_NUMBER=.*/INSTANCE_NUMBER="*XXX*"/' hana_freeze.sh hana_unfreeze.sh ，其中*XXX*为数据库实例编号。

sed -i 's/^DB_SID=.*/DB_SID="*XXX*"/' hana_freeze.sh hana_unfreeze.sh ，其中*XXX*为数据库SID。

此操作会同时修改冻结解冻脚本，所以无需再执行**步骤3**。

步骤3 执行**vi hana_unfreeze.sh** ，打开HANA示例解冻脚本，修改此脚本中的用户名、密码、实例编号与SID

hana_freeze.sh与**hana_unfreeze.sh**脚本实现了基本的数据库冻结与解冻操作，如果您在冻结、解冻时有其它额外步骤需要执行，可以自行在其中进行修改。详细说明请参见 **通过自定义脚本实现其它Linux应用的一致性备份**

⚠ 警告

冻结SAP HANA数据库时，按照SAP官方建议，需要冻结Data卷的XFS文件系统，否则可能出现数据不一致的问题。在此示例脚本中，将会查询出HANA使用的Data卷挂载点，并用xfs_freeze 命令进行冻结。

如果HANA系统未按照SAP官方建议使用一个独立分区来存放Data卷数据，而是与系统卷共用一个分区，则请修改**hana_freeze.sh**脚本，注释掉xfs_freeze相关行，防止整个系统都被冻结，但此时可能出现备份数据不一致的问题。

----结束

1.3 通过自定义脚本实现其它 Linux 应用的一致性备份

场景介绍

在Linux下，如果有其它应用需要一致性备份，可以编写自己的冻结、解冻脚本，来实现应用的保护。自定义脚本需放置在**/home/rdadmin/Agent/bin/thirdparty/ebk_user**目录中，供Agent在备份过程中调用。

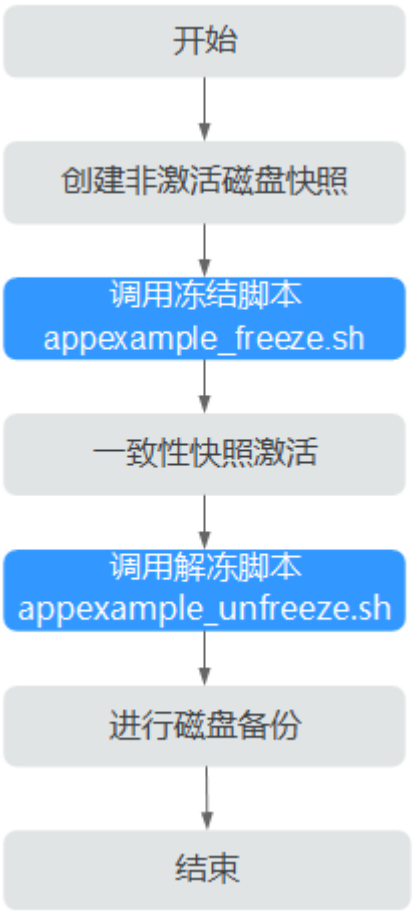
下面以一个虚构的应用appexample为例，来进行说明。

appexample是一款新的数据库，它对外提供了appexample -freeze与appexample -unfreeze两个命令来实现冻结与解冻。

用户需要开发自己的appexample_freeze.sh与appexample_unfreeze.sh脚本，供备份Agent调用以实现一致性备份。在备份过程中，会先调用appexample_freeze.sh脚本来冻结IO，冻结成功后，会进行磁盘的一致性快照激活，保证备份的数据是一致性的，最后再调用appexample_unfreeze.sh脚本解冻IO。

整体流程如**图1-1**所示：

图 1-1 数据库备份流程图



开发冻结脚本

appexample_freeze.sh示例如下：

```
#!/bin/sh
AGENT_ROOT_PATH=$1  #Agent程序调用脚本时，传入的的根目录，日志函数等会使用此变量，请不要改名
PID=$2 #Agent程序调用脚本时，传入的PID数字，用于结果的输出，请不要改名
. "${AGENT_ROOT_PATH}/bin/agent_func.sh"#引用脚本框架，提供了日志，加解密等功能
#结果处理函数，用于将结果写入到文件中，供脚本调用者获取返回值。
#入参 $1: 0表示成功，1表示失败
#无返回值
#RESULT_FILE在agent_func.sh中进行了定义
function ExitWithResult()
{
    Log "[INFO]:Freeze result is $1."
    echo $1 > ${RESULT_FILE}
    chmod 666 ${RESULT_FILE}
    exit $1
}
function Main()
{
    Log "********************************************************************"
    Log "[INFO]:Begin to freeze appexample."
    #查找appexample是否存在，如果appexample不存在，则返回0，退出
    #在冻结IO步骤中，Agent程序会依次调用每个冻结脚本，如果一个失败，总体就会失败。所以为了防止干扰其他程序的冻结过程，找不到appexample时，应返回0
    which appexample
    if [ $? -ne 0 ]
    then
            Log "[INFO]:appexample is not installed."
```



```
        ExitWithResult 0
    fi
    #调用实际的冻结命令
    appexample -freeze
    if [ $? -ne 0 ]
    then
        Log "[INFO]:appexample freeze failed."
        #冻结失败，记录结果并退出
        ExitWithResult 1
    fi
    Log "[INFO]:Freeze appexample success."
    #冻结成功，记录结果并退出
    ExitWithResult 0
}
Main
```

开发解冻脚本

appexample_unfreeze.sh示例如下：

```
#!/bin/sh
AGENT_ROOT_PATH=$1  #Agent程序调用脚本时，传入的的根目录，日志函数等会使用此变量，请不要改名
PID=$2 #Agent程序调用脚本时，传入的PID数字，用于结果的输出，请不要改名
. "${AGENT_ROOT_PATH}/bin/agent_func.sh"#引用脚本框架，提供了日志，加解密等功能
#结果处理函数，用于将结果写入到文件中，供脚本调用者获取返回值。
#入参 $1: 0表示成功，1表示失败
#无返回值
#RESULT_FILE在agent_func.sh中进行了定义
function ExitWithResult()
{
    Log "[INFO]:Freeze result is $1."
    echo $1 > ${RESULT_FILE}
    chmod 666 ${RESULT_FILE}
    exit $1
}
function Main()
{
    Log "*****************************************************************"
    Log "[INFO]:Begin to freeze appexample."
    #查找appexample是否存在，如果appexample不存在，则返回0，退出
    #在解冻IO步骤中，Agent程序会依次调用每个解冻脚本，如果一个失败，总体就会失败。所以为了防止干扰其他程序的解冻过程，找不到appexample时，应返回0
    which appexample
    if [ $? -ne 0 ]
    then
        Log "[INFO]:appexample is not installed."
        ExitWithResult 0
    fi
    #调用实际的解冻命令
    appexample -unfreeze
    if [ $? -ne 0 ]
    then
        Log "[INFO]:appexample freeze failed."
       #解冻失败，记录结果并退出
        ExitWithResult 1
    fi
    Log "[INFO]:Freeze appexample. success"
    #解冻成功，记录结果并退出
    ExitWithResult 0
}
Main
```

1.4 验证数据库备份结果

使用自定义脚本实现数据库备份完成后，可以通过如下操作验证数据库备份结果是否成功。

验证数据库备份结果（Linux）

下面以MY SQL数据库为例进行验证。

步骤1 登录MY SQL数据库，创建新的数据库。

步骤2 创建数据库成功后，创建存储过程，可以参考图1-2。

图 1-2 创建存储过程

```
DELIMITER //
CREATE DEFINER=`root`@`localhost` PROCEDURE `test_insert_xuwei3`()
BEGIN
declare i int;
declare v float;
set i = 0;
while i < 10000000
do
select RAND()*100 into v;
insert into xuwei1_test values(i, 'xxxxxx', now());
set i = i+1;
end while;
END
//
DELIMITER ;
```

步骤3 进入云服务器备份控制台，对目标弹性云服务器创建数据库备份，并勾选数据库备份。

步骤4 待备份完成后，进入/home/rdadmin/Agent/log/thirdparty.log，查看冻结、解冻日志，确定冻结解冻时间。

步骤5 使用新创建的数据库备份恢复目标弹性云服务器。恢复成功后，登录云服务器和数据库，查看表中最后一条插入数据对应的时间。

步骤6 对比步骤5日志显示的VSS冻结成功时间和步骤4的时间。冻结成功之前会停止插入数据，所以步骤5的时间比步骤4早。若步骤5的时间比步骤4早，则表示应用一致性备份成功。

----结束

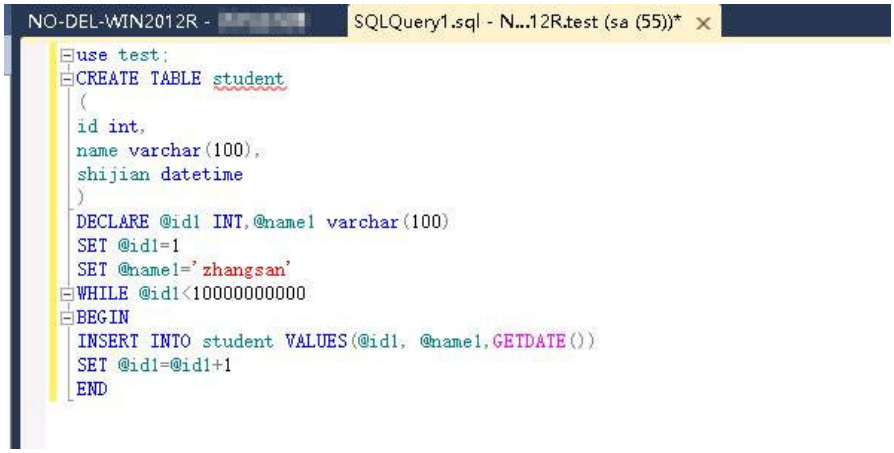
验证数据库备份结果（Windows）

下面以SQL_SERVER数据库为例进行验证。

步骤1 登录SQL_SERVER数据库，创建新的数据库。

步骤2 创建数据库成功后，创建存储过程，可以参考图1-3。

图 1-3 创建存储过程



步骤3 进入云备份控制台，对目标弹性云服务器创建数据库备份，并勾选数据库备份。

步骤4 待备份完成后，进入Cloud Server Backup Agent-WIN64\log\ rdagent.txt文件，查看冻结、解冻日志，确定冻结解冻时间。如图中所示的17:28:51。

图 1-4 查看日志

```
[2018-11-14 17:28:46][0x0000531600001536][2052][SYSTEM][INFO][Requester.cpp,1369]Start snap shot set.
[2018-11-14 17:28:46][0x0000531600001536][2052][SYSTEM][INFO][Requester.cpp,1372]Add to snapshot set.
[2018-11-14 17:28:46][0x0000531600001536][2052][SYSTEM][INFO][Requester.cpp,1375]Prepare for backup.
[2018-11-14 17:28:46][0x0000531600001535][2052][SYSTEM][INFO][Requester.cpp,1261]Begin prepare for backup.
[2018-11-14 17:28:46][0x0000531600001535][2052][SYSTEM][INFO][Requester.cpp,1272]Prepare for backup succ.
[2018-11-14 17:28:46][0x0000531600001536][2052][SYSTEM][INFO][Requester.cpp,1378]Do snapshot set.
[2018-11-14 17:28:46][0x0000531600001535][2052][SYSTEM][INFO][Requester.cpp,1278]Begin create the shadow (Do SnapShot Set).
[2018-11-14 17:28:51][0x0000531600001535][2052][SYSTEM][INFO][Requester.cpp,1317]Create the shadow (Do SnapShot Set) succ.
[2018-11-14 17:28:51][0x0000531600001536][2052][SYSTEM][INFO][Requester.cpp,227]Freeze volume succ.
[2018-11-14 17:28:51][0x0000531600001536][2052][SYSTEM][INFO][Requester.cpp,180]Freeze file sys, succ.
[2018-11-14 17:28:51][0x0000531600001536][2052][SYSTEM][INFO][App.cpp,383]Vss freeze success.
[2018-11-14 17:28:51][0x0000531600001536][2052][SYSTEM][INFO][AppPlugin.cpp,157]Freeze app succ.
[2018-11-14 17:28:51][0x0000531600001536][4872][SYSTEM][INFO][MessageProcess.cpp,1034]Json key "loop_time" does not exist.
[2018-11-14 17:28:51][0x0000531600001536][4872][SYSTEM][INFO][FTExceptionHandle.cpp,849]Update monitor obj freeze begin time
[2018-11-14 17:28:52][0x0000531600001536][544][SYSTEM][INFO][Communication.cpp,400]End accept fcgx
[2018-11-14 17:28:52][0x0000531600001536][544][SYSTEM][INFO][Authentication.cpp,104]strClientCertDN: CN=BCManager eBackup Cl
[2018-11-14 17:28:52][0x0000531600001536][544][SYSTEM][INFO][Authentication.cpp,130]Client IP address 100.125.1.142 Auth suc
[2018-11-14 17:28:52][0x0000531600001536][544][SYSTEM][INFO][Communication.cpp,390]Begin accept fcgx
[2018-11-14 17:28:53][0x0000531600001536][2052][SYSTEM][INFO][AppPlugin.cpp,168]Begin unfreeze app.
[2018-11-14 17:28:53][0x0000531600001536][2052][SYSTEM][INFO][App.cpp,392]Begin vss unfreeze.
[2018-11-14 17:28:53][0x0000531600001536][2052][SYSTEM][INFO][Requester.cpp,275]Begin unfreeze all.
[2018-11-14 17:28:53][0x0000531600001536][2052][SYSTEM][INFO][Requester.cpp,1703]Begin wait for async ex.
[2018-11-14 17:28:53][0x0000531600001536][2052][SYSTEM][INFO][Requester.cpp,1733]End wait for async ex, return 0x0004230a (V
[2018-11-14 17:28:53][0x0000531600001536][2052][SYSTEM][INFO][Requester.cpp,1579]VSS async finished.
[2018-11-14 17:28:53][0x0000531600001536][2052][SYSTEM][INFO][Requester.cpp,303]End unfreeze all.
[2018-11-14 17:28:53][0x0000531600001536][2052][SYSTEM][INFO][App.cpp,415]VSS unfreeze success.
[2018-11-14 17:28:53][0x0000531600001536][2052][SYSTEM][INFO][App.cpp,424]Begin vss endbakup.
[2018-11-14 17:28:53][0x0000531600001536][2052][SYSTEM][INFO][Requester.cpp,311]Begin end bakcup.
[2018-11-14 17:29:05][0x0000531600001536][2052][SYSTEM][INFO][Requester.cpp,333]End end backup.
[2018-11-14 17:29:05][0x0000531600001536][2052][SYSTEM][INFO][App.cpp,445]Vss endbakup success.
[2018-11-14 17:29:05][0x0000531600001536][2052][SYSTEM][INFO][App.cpp,342]Unfreeze all apps success.
[2018-11-14 17:29:05][0x0000531600001536][2052][SYSTEM][INFO][AppPlugin.cpp,185]Unfreeze app succ.
[2018-11-14 17:29:05][0x0000531600001536][4872][SYSTEM][INFO][MessageProcess.cpp,1034]Json key "loop_time" does not exist.
```

步骤5 使用新创建的数据库备份恢复目标弹性云服务器。恢复成功后，登录云服务器和数据库，查看表中最后一条插入数据对应的时间(17:28:49)的记录。

步骤6 对比**步骤5**日志显示的VSS冻结成功时间和**步骤4**的时间。冻结成功之前会停止插入数据，所以**步骤5**的时间比**步骤4**早。若**步骤5**的时间比**步骤4**早，则表示应用一致性备份成功。

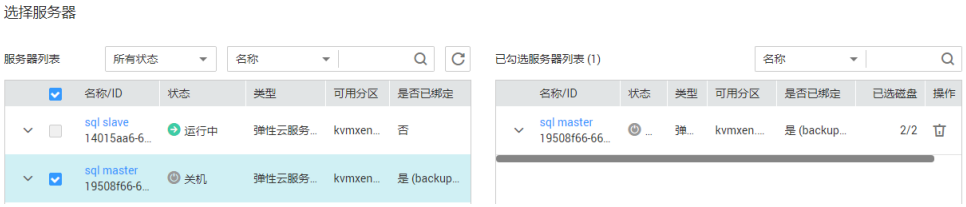
----结束

1.5 保护 SQL Server

保护 Failover Cluster 模式下的 SQL Server

当前云服务器备份只支持单个虚拟机的一致性备份，对于集群数据库暂不支持，完整支持将在后续版本中推出。

在Failover Cluster模式下，SQL Server服务只在主节点上是启动的，故在创建云服务器备份时，只需要将主节点加入策略进行备份。在主备发生切换后，及时调整策略，确保始终对主节点进行备份。在恢复时，请先停止所有备节点，然后还原主节点。



保护 Always on Availability Groups 模式下的 SQL Server

当前云服务器备份只支持单个虚拟机的一致性备份，对于集群数据库暂不支持，完整支持将在后续版本中推出。

在Always On模式下，SQL Server服务在主备节点上都是启动的，数据由主复制到备，主上拥有全部的数据。故在创建云服务器备份时，只需要将主节点加入策略进行备份。在主备发生切换后，及时调整策略，确保始终对主节点进行备份。

由于SQL Server自身的机制，在恢复主时，可能会触发同步，使备节点上的数据也被覆盖，导致备份时刻之后新产生的数据丢失，所以建议只有在主备节点均不可用时才进行整机恢复，防止非预期的数据丢失。

1.6 自定义脚本问题定位方法

如果自定义脚本存在缺陷，可能导致数据库备份失败，此时可以打开**/home/rdadmin/Agent/log/thirdparty.log**，查看日志进行定位。

图1-5为一个冻结MySQL数据库失败时的日志样例

图 1-5 日志示例

```
18-09-13--22:30:10:[30243][root] **********************************************************************
18-09-13--22:30:10:[30243][root] [INFO]:Begin to freeze mysql.
Id  User    Host    db  Command Time    State   Info
20  root    localhost   test123 Sleep   1063        NULL
21  root    localhost   test123 Sleep   1066        NULL
24  root    localhost   NULL    Query   23  User sleep  select 1 and sleep(60)
27  root    localhost   NULL    Query   0   NULL    show processlist
18-09-13--22:30:10:[30243][root] [ERROR]:MySQL already been freezed
18-09-13--22:30:10:[30243][root] [INFO]:mysql freeze result is 1.
18-09-14--10:07:54:[7162][root] **********************************************************************
```

第一列 **18-09-13--22:30:10** 为日志记录时间

第二列 **[30243]** 为脚本的PID编号

第三列 **[root]** 为脚本的执行用户

第四列 **[INFO]** 或 **[ERROR]** 为日志级别

一般脚本调用失败时，打开日志文件，找到相应时间点的ERROR即可初步确定问题原因。例如**图1-5**中的错误就是因为MySQL已经处于冻结状态，再次冻结，就会出错。

2 通过数据备份开展定期恢复演练

背景

根据数据安全法，需要对数据容灾备份**定期开展数据恢复测试**。恢复是指利用备份软件把所备份的数据内容恢复到数据源。由于业务系统日常运行过程中，经常无法直接在所备份的服务器进行真实环境恢复操作。但为了验证备份数据的可用性以及备份方案完整性、可靠性以及应对未来系统突发事件发生，可以通过备份新建资源的方式来对备份介质以及备份方案进行检验，来确保备份的可恢复验证。

演练原则

- 参考《CBR云备份备份策略最佳实践》里**灾备策略**里的演练频率，有计划、周期性地对备份数据进行恢复演练。
- 以备份资源为单位在该资源的所有备份内抽样随机进行，不必每个备份都进行，但要保证在一定期限内每种资源的每类备份至少有一次备份被恢复验证过。
- 为防止干扰实际业务，恢复演练以使用备份创建新资源实例进行，禁止直接恢复源实例。
- 下发备份恢复任务后，恢复任务成功，备份能够正常恢复资源，且恢复的数据与原来一致，正确性与预期匹配，则视为恢复成功。
- 下发备份恢复任务后，如果恢复任务失败，或者恢复任务成功，但数据存在丢失、无法读取的情况，则视为恢复失败，请及时联系华为云工程师进行定位处理。
- 操作员应详细记录演练的周期、过程及结果。

资源与成本

表 2-1 资源和成本规划

资源	资源说明	数量	每月费用
云服务器备份存储库	存储库容量大于等于所需要备份云服务器资源的容量总和	1	具体的计费方式及标准请参考**计费说明**。

SFS Turbo备份存储库	存储库容量大于等于所需要备份SFS Turbo资源的容量	1	
弹性云服务器	与待演练服务器的配置相同	1	
SFS Turbo文件系统	与待演练的SFS Turbo文件系统大小相同	1	
RDS数据库实例	与待演练的RDS数据库配置相同	1	

备份演练

三种备份类型的备份演练操作流程如下：

云服务器备份演练

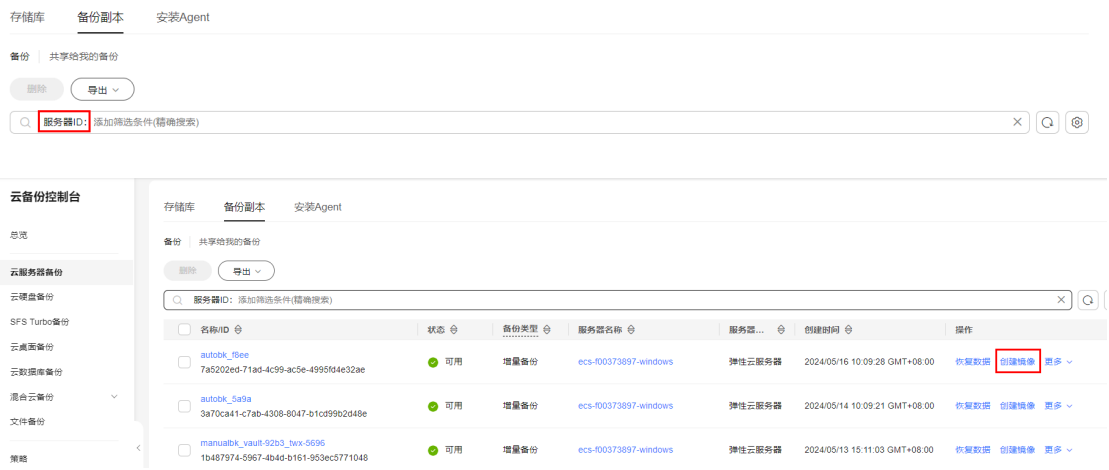
步骤1 以核心系统的某个云服务器为例，每月进行备份恢复演练任务。

步骤2 登录云备份管理控制台。

1. 登录管理控制台。
2. 单击管理控制台左上角的，选择区域。
3. 单击“”，选择“存储 > 云备份”。

步骤3 选择“云服务器备份”页签，选择“备份副本”。

步骤4 获取该云服务器的备份列表：选择“服务器ID”，输入需要验证的云服务器ID，单击“搜索”。



步骤5 选择任意一个备份副本，单击“创建镜像”，进入IMS创建私有镜像界面，输入镜像名称后，单击“确定”。

步骤6 待镜像创建完成后，使用该镜像申请云服务器。

步骤7 将新创建的云服务器与当前云服务器进行对比，查看数据是否一致，符合预期，观察业务是否正常。

----结束

云文件系统备份演练

步骤1 以核心系统的某个云文件系统为例，每月进行备份恢复演练任务。

步骤2 登录云备份管理控制台。

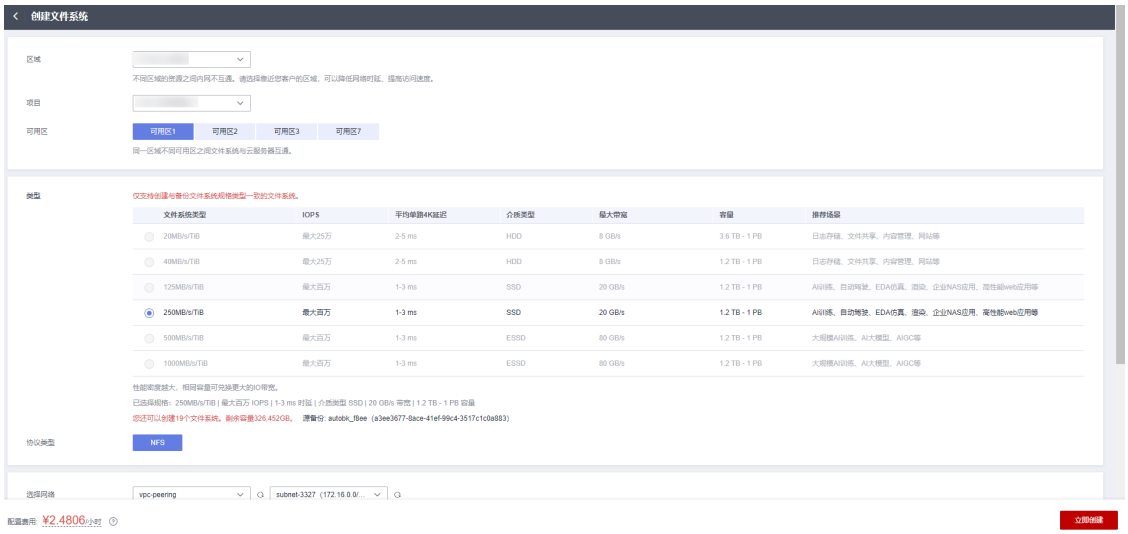
1. **登录管理控制台**。
2. 单击管理控制台左上角的 ，选择区域。
3. 单击“ ”，选择“存储 > 云备份”。

步骤3 选择“SFS Turbo备份”签页，选择“备份副本”。

步骤4 获取该云文件系统的备份列表：选择“文件系统ID”，输入需要验证的文件系统ID，单击“搜索”。

步骤5 选择需要恢复的备份，单击“创建文件系统”，并跳转到文件系统创建页。

步骤6 选择云文件系统创建参数，确认提交，完成创建新的文件系统。



步骤7 查看比对新创建的云文件系统数据，校验数据是否符合预期。

----结束

云数据库备份演练

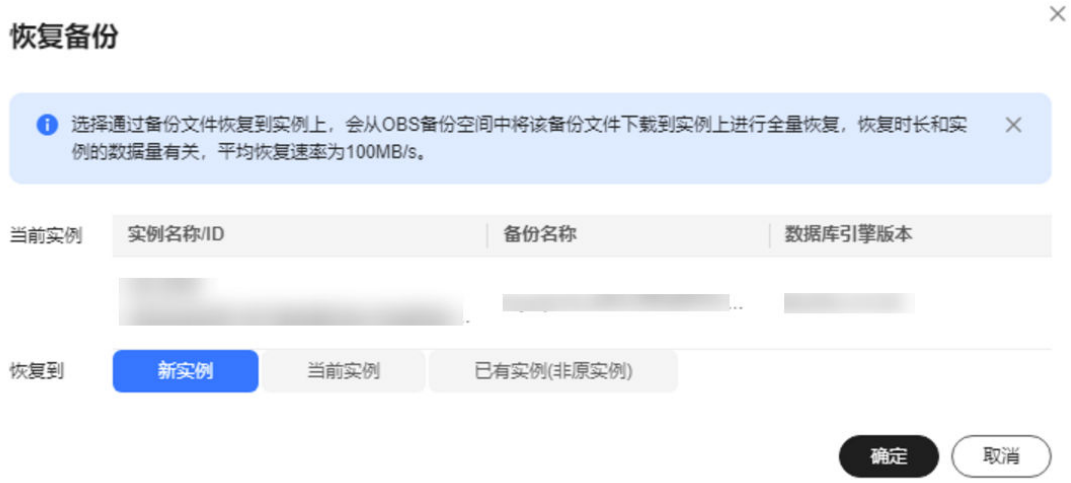
步骤1 登录云备份管理控制台。

1. **登录管理控制台**。
2. 单击管理控制台左上角的 ，选择区域。

3. 单击“ ☰ ”，选择“数据库 > 云数据库 RDS”。

步骤2 单击“备份管理”，选择需要恢复的备份，单击操作列的“恢复”。

步骤3 选择恢复到“新实例”，单击“确定”，进入“恢复到新实例”的服务选型页面。



步骤4 新实例的数据库引擎和数据库版本，自动与原实例相同。存储空间大小默认和原实例相同，且必须大于或等于原实例存储空间大小。其余参数详情可参考**帮助指导**。

步骤5 单击“立即购买”，按照指导购买新数据库，等待数据库实例状态由“创建中”变为“正常”，说明恢复成功。

步骤6 恢复成功后登录数据库实例进行验证，校验数据是否匹配预期。

----结束

3 通过业务分级制定最佳备份策略

背景

资源的备份策略决定了备份周期与保留规则，实践中需要根据数据的重要程度、用户云上部署的业务系统的不同分级，配置不同的数据备份策略。

资源与成本规划

表 3-1 资源和成本规划

资源	资源说明	数量	每月费用
备份存储库	创建备份策略后，需要绑定备份存储库，以实现按照创建的备份策略进行定时备份	1	具体的计费方式及标准请参考**计费说明**。

业务分级

同时结合业务实际，按照业务系统重要性及业务系统中断对整个公司业务影响的范围和程度，可考虑将业务系统分为三类：

- **核心系统**
  a. 支撑公司核心业务流程，一旦停止服务会给公司运营造成重大影响或重大财务损失，如购买系统；
  b. 支撑公司关键应用的重要基础设施和办公系统，一旦中断，导致大量员工正常工作瘫痪。
- **重要系统**
  a. 支撑公司重要的业务流程，一旦停止服务会给公司业务运营造成严重影响，或造成严重财务损失；
  b. 支撑公司重要应用的基础设施和办公系统，一旦中断，会严重影响员工的正常办公。
- **一般系统**
  a. 支撑公司业务流程，一旦停止服务对公司运营或财务造成损失，如培训系统；

b. 支撑公司普通应用的基础设施或办公系统，一旦中断，会影响员工的正常办公。

一般情况下，可根据业务系统影响评估数据等级，核心数据对应于核心系统，重要系统对应于重要数据，一般系统对应一般数据。如有数据分类与业务系统类别不匹配的情况，则业务系统向更高级别系统定级。例如业务系统被评估为重要系统，但是数据类型属于数据安全法的核心数据，则该系统向上一级定义为核心系统。

灾备策略

按照用户云上部署的业务系统的不同分级，建议配置不同的数据备份策略，针对常见的云内资源采用如下数据备份策略，该策略可根据实际业务情况自行调整。参照策略管理章节**创建备份策略**。

系统分类	备份对象	RPO	保留时间	全量备份	增量备份	异地备份	演练频率
核心系统	云服务器	4h	1年以上	每周	6次/日	是	月
	云数据库	4h	1年以上	每周	6次/日	是	月
	云文件系统	4h	1年以上	每周	6次/日	是	月
重要系统	云服务器	12h	1年	双周	2次/日	是	季度
	云数据库	12h	1年	双周	2次/日	是	季度
	云文件系统	12h	1年	双周	2次/日	是	季度
一般系统	云服务器	24h	6个月	每月	1次/日	否	半年
	云数据库	24h	6个月	每月	1次/日	否	半年
	云文件系统	24h	6个月	每月	1次/日	否	半年

4 通过 CBR+HSS 搭建云上备份防勒索系统

应用场景

勒索病毒，也称为勒索软件，是一种特殊的恶意软件，与其他病毒最大的不同在于攻击手段伴随着有组织的网络威胁攻击和加密数据后勒索赎金。

勒索病毒已成为全球主要网络威胁，严重影响着数字经济的发展，而面对勒索病毒的侵害，大多数企业并没有全面和有效的方法予以应对。不得不通过支付高额的赎金来请求黑客对加密的数据进行解锁。而在企业遭受到攻击的同时，会造成企业的重要数据丢失、企业停工停产、合同违约、商誉减值、企业管理者离职以及众多不可预计的后果。

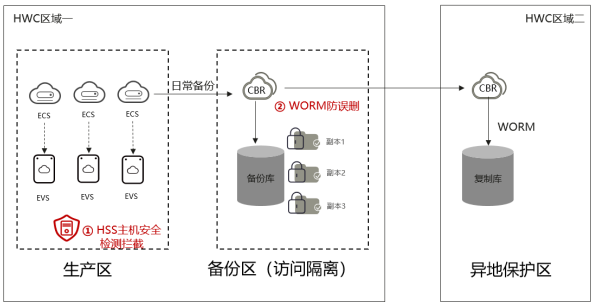
华为云备份防勒索解决方案与主机安全服务HSS结合，为客户提供有效云上防勒索解决方案，满足客户对数据安全保护的需求。

介绍视频

方案架构

云备份所提供的防勒索解决方案服务，从业务实际出发，以客户数据安全要求为导向，帮助客户搭建一个安全可靠的备份防勒索系统，满足客户对于勒索病毒防护的需求，保护数据资产，尽可能减小损失。通过存储库天然具备的访问隔离属性、WORM等特性，实现当云服务器发生勒索攻击时，能提供至少一份干净可用、不被篡改的数据用于安全恢复，提升数据安全的韧性能力，满足客户对于勒索病毒防护的需求。

图 4-1 基于 HSS+CBR 的云上备份防勒索解决方案



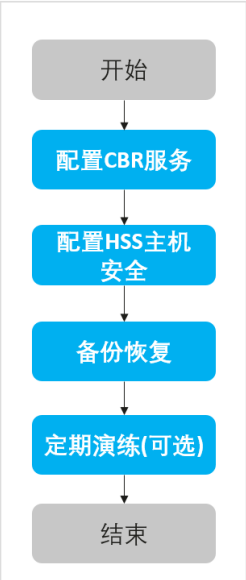
方案优势

- HSS主机安全服务与CBR云备份服务联动，当HSS检测到病毒入侵后立即触发CBR备份，有效减小数据损失；
- CBR备份存储库天然具备隔离属性，结合WORM能力，实现本地备份的防篡改防误删；
- 将CBR备份复制到其他区域，结合WORM能力，实现异地备份的防篡改防误删。

操作流程

本章节介绍云备份防勒索解决方案的操作流程如图4-2所示。

图 4-2 云上备份防勒索方案操作流程



配置 CBR 备份并开启备份锁定

购买云服务器备份存储库

备份存储库是CBR服务的基本单元，使用CBR服务前必须要先购买备份存储库。如果使用量超过购买的容量后将不能再进行备份，因此购买存储库时建议购买自动扩容存储库，避免因为容量问题导致备份失败。

步骤1 登录云备份管理控制台。

1. 登录管理控制台。
2. 单击管理控制台左上角的，选择区域。
3. 单击“ ”，选择“存储 > 云备份 CBR”。选择对应的备份目录。

步骤2 在界面右上角单击“购买云服务器备份存储库”。

步骤3 选择计费模式。

- 包年包月是预付费模式，按订单的购买周期计费，适用于可预估资源使用周期的场景，价格比按需计费模式更优惠。

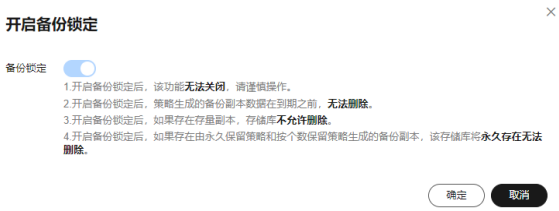
- 按需计费是后付费模式，根据实际使用量进行计费，可以随时购买或删除存储库。费用直接从账户余额中扣除。

  说明

  如果选择的是按需计费的存储库，则自动扩容建议选择“立即配置”，避免因为容量问题导致备份失败。

步骤4 选择是否启用备份锁定，也可以后续选择开启。

开启备份锁定后，可以避免备份被误删除或者恶意删除，提升数据安全性。



须知

- 开启备份锁定后，该功能无法关闭，策略生成的备份只支持过期自动删除，存在存量副本的存储库不允许删除。手动创建的备份不受备份锁定的约束，支持手动删除。
- 请详细阅读说明并确认是否开启，建议开启。
- 在购买存储库时若没有选择启用备份锁定，可以在任一备份页面，找到目标存储库，单击存储库所在列的“更多 > 开启备份锁定”，在备份锁定弹框中，打开备份锁定开关。更多详细内容请参见**开启备份锁定**。

步骤5 其余选项按需配置，具体操作可参见**购买云服务器备份存储库**。

存储库购买完成后，可以进行备份策略或复制策略的创建、修改、删除、绑定至存储库等操作。具体操作请参照**策略管理**章节。

----结束

配置 HSS 主机安全

本最佳实践方案需要使用到HSS主机安全服务，详情请参考**HSS防勒索最佳实践**。

备份恢复

当遭受勒索病毒攻击后，当前尚无有效工具能够破解勒索病毒，唯一的方案只能通过备份快速恢复数据。通过备份恢复云服务器时，首先判断该备份是否已被勒索病毒感染，如果备份正常，则参考**使用云服务器备份恢复数据**；如果备份为勒索加密备份，则建议的备份恢复操作如下：

步骤1 用**IAM管理员账号**登录**登录管理控制台**。

步骤2 单击管理控制台左上角的 ，选择区域。

步骤3 在控制台的左侧导航窗格中选择“计算 > 弹性云服务器 ECS”，进入弹性云服务器界面。

步骤4 按照**通过规格选型引导购买云服务器**在**独立的VPC内**购买弹性云服务器。

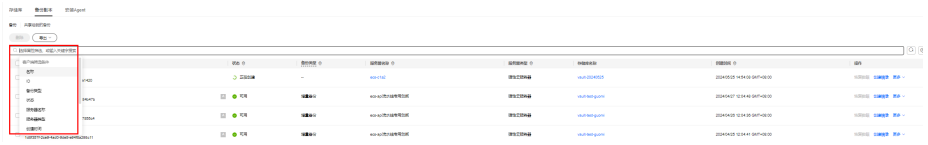
说明

此处使用独立VPC，防止勒索病毒在干净区域内感染。

步骤5 在IAM控制台的左侧导航窗格中选择“存储 > 云备份 CBR”，进入云备份服务界面。

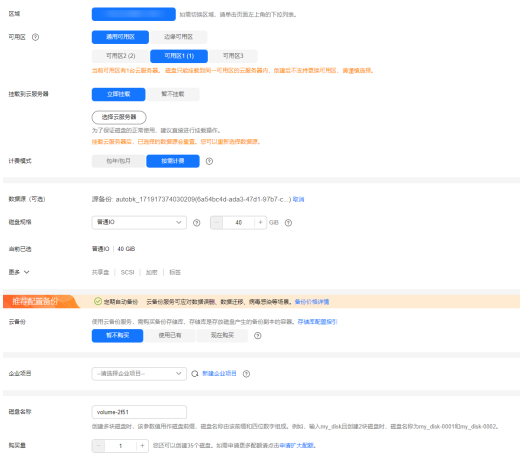
步骤6 在左侧导航栏单击“云服务器备份 > 备份副本”，在备份列表上方，可以通过备份名称、存储库ID、服务器名称、服务器ID等条件搜索，搜索需要恢复的云服务器备份。

图 4-3 筛选备份



步骤7 单击选中的云服务器备份，进入详情页后，单击“磁盘级备份”。

步骤8 单击“创建磁盘”，进入到“购买磁盘”界面，按需创建磁盘。



步骤9 待磁盘创建成功后，将磁盘挂载到**步骤3**创建的虚拟机上，登录虚拟机查看磁盘下的数据是否正常。

步骤10 步骤9中正常的未加密的数据，可以通过对象存储obsutil工具上传到对象存储桶里。

在OBS**管理控制台**左侧导航栏选择“桶列表”，在页面右上角单击“创建桶”，系统弹出如下所示的页面，创建临时桶ransomware_anti_bucket_tmp，用于临时保存数据。具体操作可参见**创建桶**。

图 4-4 创建桶

步骤11 在控制台的左侧导航窗格中选择“管理与监管 > 统一身份认证服务 IAM”，进入统一身份认证服务界面。

步骤12 单击“权限管理 > 权限 > 创建自定义策略”，策略名称填写ransomware_anti_access，策略配置方式选择json视图，填写权限如下：

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "obs:object:GetObject",
                "obs:bucket:HeadBucket",
                "obs:object:GetObjectAcl",
                "obs:object:PutObject"
            ],
            "Resource": [
                "OBS:*:*:object:*",
                "OBS:*:*:bucket:ransomware_anti_bucket_tmp"
            ]
        }
    ]
}
```

单击“确定”。

步骤13 单击“用户组 > 创建用户组”，输入用户组名“ransomware_anti_group”,单击确定后，在该用户组的操作栏里单击“授权”，选择策略“ransomware_anti_access”后，确定。

步骤14 单击“创建用户”，输入用户名ransomware_anti_user，访问方式选择“编程访问”，凭证类型选择“访问密钥”，单击“下一步”，选择用户组“ransomware_anti_group”后，“创建用户”创建子用户。

步骤15 创建成功后，获取用户的AK/SK信息，参考**obsutil上传对象**，将文件上传到临时桶里后，下载即可。

步骤16 文件传输完成后，删除虚拟机，删除临时用户、权限组和临时桶。

----结束

周期性演练（可选）

根据数据安全法，需要对数据容灾备份**定期开展数据恢复测试**。恢复是指利用备份软件把所备份的数据内容恢复到数据源。由于业务系统日常运行过程中，经常无法直接在所备份的服务器进行真实环境恢复操作。但为了验证备份数据的可用性以及备份方案完整性、可靠性以及应对未来系统突发事件发生，可以通过备份新建资源的方式来对备份介质以及备份方案进行检验，来确保备份的可恢复验证。

通过数据备份可开展定期恢复演练，具体步骤可参见**云服务器备份演练**。

5 通过 VMware 备份主机批量恢复脚本

场景介绍

针对VMware备份上云场景，云备份（Cloud Backup and Recovery，CBR）增加VMware版本兼容性。但是，VMware混合云备份界面只能操作单个虚拟机进行备份数据恢复，在虚拟机较多时，界面操作步骤繁琐且并发太少。通过该章节内容实现脚本批量执行备份数据恢复，以增加备份副本恢复的并发数，提高效率。

脚本说明

脚本基于Python语言开发，主要实现备份数据批量恢复功能、批量回滚功能等。

config.py	配置文件
main.py	备份数据批量恢复主流程
rollback.py	批量回滚，用于批量删除云服务器

前提条件

- 熟悉Python语言，并有Python环境搭建基础。
- 熟悉华为云备份数据恢复功能。

方案使用到的接口

URL	所属服务	用途	API文档
POST /v3/auth/tokens	IAM	认证鉴权	认证鉴权
GET /v3/{project_id}/backups/{backup_id}	CBR	查询备份详情	查询指定备份

URL	所属服务	用途	API文档
POST /v1/{project_id}/cloudservers	ECS	创建ECS	创建云服务器
GET /v1/{project_id}/jobs/{job_id}	ECS	查询ECS是否创建完成	查询任务的执行状态
GET /v1/{project_id}/cloudservers/{server_id}	ECS	查询ECS详情，获取创建好的虚拟机的挂载的磁盘信息	查询服务器详情
POST /v3/{project_id}/backups/{backup_id}/restore	CBR	使用备份恢复数据	备份恢复

参数获取

用户需要收集配置数据，完成config.py中变量的初始赋值，主要包含IAM鉴权参数、项目公共参数、服务器参数、备份数据参数、回滚参数以及监控任务参数。

IAM鉴权参数

1. 登录管理控制台。
2. 在“控制台”页面，鼠标移动至右上方的用户名，在下拉列表中选择“我的凭证”。
3. 在“我的凭证”界面，单击API凭证界面查看。

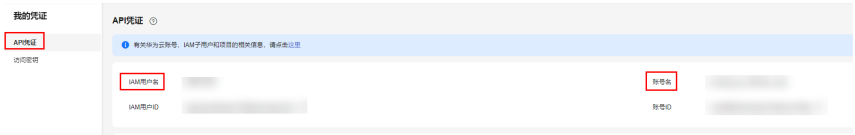


表 5-1 IAM 鉴权参数说明

参数	是否必选	参数类型	描述
iamDomain	是	String	IAM用户名
iamUser	是	String	IAM用户所属账号名
iamPassword	是	String	IAM用户密码

公共参数

1. 登录管理控制台。
2. 在“控制台”页面，鼠标移动至右上方的用户名，在下拉列表中选择“我的凭证”。

3. 在“我的凭证”界面，单击API凭证界面查看。

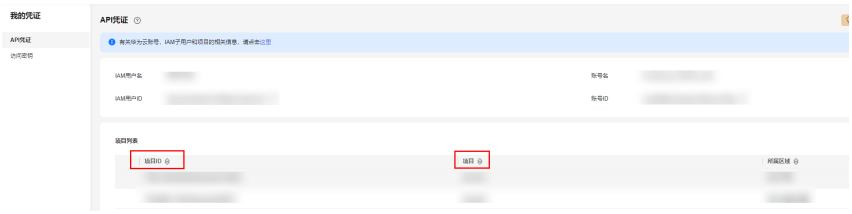


表 5-2 公共参数说明

参数	是否必选	参数类型	描述
projectName	是	String	项目
projectId	是	String	项目ID

服务器参数

- **获取imageRef参数**
  a. 登录云服务器管理控制台。
     i. **登录管理控制台**。
     ii. 单击管理控制台左上角的，选择区域。
     iii. 单击“ ”，选择“计算 > 弹性云服务器”。
  b. 在左侧导航栏选择“镜像服务”，找到对应镜像并复制镜像ID。

- **获取flavorRef参数**
  在弹性云服务器控制台界面，单击“购买弹性云服务器”，在“规格”项中找到对应的待创建云服务器的系统规格ID。

- **获取vpcId参数**

  a. 登录云服务器管理控制台。

  i. **登录管理控制台**。

  ii. 单击管理控制台左上角的，选择区域。

  iii. 单击“”，选择“网络 > 虚拟私有云”。

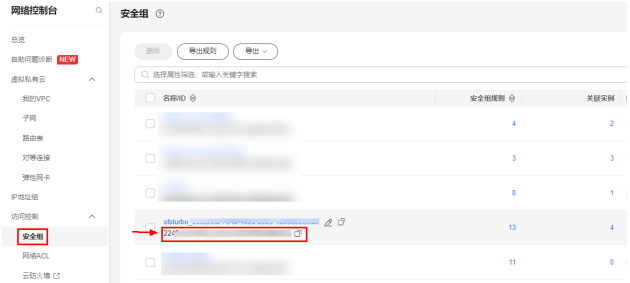
  b. 在左侧导航栏选择“虚拟私有云 > 我的VPC”，找到对应vpcid。

- **获取subnetId参数**

  在网络控制台界面，左侧导航栏选择“虚拟私有云 > 子网”，找到对应VPC下已创建的子网（subnet）的网络ID。

- **获取securityGroups参数**

  在网络控制台界面，左侧导航栏选择“访问控制 > 安全组”，找到云服务器对应安全组信息。



- **获取volumeType参数**

  云硬盘类型参数，目前支持"SATA"，"SAS"，"GPSSD"，"SSD"，"ESSD"，"GPSSD2"，"ESSD2"七种。

  具体信息可参见**创建云硬盘**volume_type参数说明。

- **获取enterpriseProjectId参数**

  企业项目ID，默认为'0'。

- **获取keyName参数**

  密钥名称，默认不需要配置。如果需要使用SSH密钥方式登录云服务器，请指定已创建密钥的名称。

表 5-3 服务器参数说明

参数	是否必选	参数类型	描述
imageRef	是	String	待创建云服务器的系统镜像，需要指定已创建镜像的ID，ID格式为通用唯一识别码（Universally Unique Identifier，简称UUID）。
volumeType	是	String	云服务器系统盘对应的磁盘类型，需要与系统所提供的磁盘类型相匹配。 默认值："SSD"
flavorRef	是	String	待创建云服务器的系统规格的ID。 示例： "s7n.small.1"
vpcId	是	String	待创建云服务器所属虚拟私有云（简称VPC），需要指定已创建VPC的ID，UUID格式。
subnetId	是	String	待创建云服务器所在的子网信息。 需要指定vpcid对应VPC下已创建的子网（subnet）的网络ID，UUID格式。
enterpriseProjectId	是	Integer	企业项目ID。 默认值：0
securityGroups	是	List	云服务器对应安全组信息。 ["xxx", "xxx"]
keyName	否	String	如果需要使用SSH密钥方式登录云服务器，请指定已创建密钥的名称。

备份数据参数

1. **登录管理控制台**。
2. 单击管理控制台左上角的 ，选择区域。
3. 单击“ ”，选择“存储 > 云备份 CBR”。

4. 选择“混合云备份 > VMware备份”，在备份副本列表获取备份数据ID。

表 5-4 备份数据参数说明

参数	是否必选	参数类型	描述
backupIds	是	List	备份数据ID列表。

回滚参数

执行备份数据恢复脚本后，待数据恢复完成，通过返回的日志中的serverId获取云服务器ID列表。


```
{
  "successJobList": [
    {
      "backupId": "0753ffd5-5b2f-4acb-a055-9446cb265e43",
      "serverId": "09364fb9-7cf8-405b-b174-69be603b6258",
      "status": "SUCCESS"
    },
    {
      "backupId": "4534750e-268e-4d64-90aa-b76dd703f484",
      "serverId": "0bae713d-0159-44a0-ba58-790667f0febb",
      "status": "SUCCESS"
    }
  ],
  "failedJobList": [
    {
      "backupId": "f85e9978-4eef-4707-a73c-8015635c2af2",
      "status": "FAILED",
      "msg": "create server error"
    },
    {
      "backupId": "2440b565-cff1-47f6-85b1-22199e0882a3",
      "status": "FAILED",
      "msg": "create server error"
    }
  ]
}
```


表 5-5 回滚参数说明

参数	是否必选	参数类型	描述
serverIds	是	List	云服务器ID列表。 备份数据恢复完成后获取。
deletePublicip	否	Boolean	配置删除云服务器是否删除云服务器绑定的弹性公网IP。 默认：False

参数	是否必选	参数类型	描述
deleteVolume	否	Boolean	配置删除云服务器是否删除云服务器对应的数据盘。 默认：True

监控任务参数

表 5-6 监控任务参数说明

参数	是否必选	参数类型	描述
delayInSeconds	是	Integer	循环监控任务状态时间间隔，单位为秒。 默认：10

备份数据恢复操作

前提条件

- 已准备好备份数据。
- 完成config.py脚本中的配置项赋值。

  config.py示例如下：

```
# IAM鉴权参数
iamDomain = ""
iamUser = ""
iamPassword = ""
# 项目参数
projectName = ""
projectId = ""
# 服务器参数
# CentOS 7.9 64bit
imageRef = ""
volumeType = "SSD"
flavorRef = "s7n.small.1"
vpcId = ""
subnetId = ""
enterpriseProjectId = 0
securityGroups = [
    ""
]
keyName = ""
# 备份数据参数
backupIds = ["",
             "",
             "",
             ""
             ]
# 备份数据回退参数
serverIds = ["", ""]
deletePublicip = False
deleteVolume = True
# 监控任务参数
delayInSeconds = 10
```


执行备份数据恢复脚本

通过命令python .\main.py执行main.py文件，开始备份数据恢复主流程。main.py示例如下：

```
import requests
import json
import time
import config


class IamLogin:
    """
    login by iam, return token
    """

    def __init__(self, iamDomain: str, iamUser: str, iamPassword: str, projectName: str):
        self.iamDomain = iamDomain
        self.iamUser = iamUser
        self.iamPassword = iamPassword
        self.projectName = projectName
        self.iamTokenUrl = "https://iam.{}.myhuaweicloud.com/v3/auth/tokens".format(self.projectName)

    def get_auth_token(self) -> dict:
        """
        return token
        """
        payload = {
            "auth": {
                "identity": {
                    "methods": [
                        "password"
                    ],
                    "password": {
                        "user": {
                            "domain": {
                                "name": self.iamDomain
                            },
                            "name": self.iamUser,
                            "password": self.iamPassword
                        }
                    }
                },
                "scope": {
                    "project": {
                        "name": self.projectName
                    }
                }
            }
        }
        json_data = json.dumps(payload)
        try:
            response = requests.post(self.iamTokenUrl,
                                     data=json_data,
                                     headers={'Content-Type': 'application/json'},
                                     verify=False)
            # 检查请求是否成功
            response.raise_for_status()
            # 获取响应头的token
            access_token = response.headers.get("X-Subject-Token")
            return {
                "token": access_token,
                "code": 0
            }
        except requests.exceptions.RequestException as e:
            print("get token error: ", e)
            return {
                "code": 1,
                "msg": "get token error"
```

```
        }

class GetBackup:
    """
    获取Backup信息
    """

    def __init__(self, token, projectName, projectId, backupId) -> None:
        self.token = token
        self.header = {
            'Content-Type': 'application/json', 'X-Auth-Token': self.token
        }
        self.projectName = projectName
        self.projectId = projectId
        self.backupId = backupId

    def get_backup(self) -> dict:
        """
        根据id查询备份数据
        """
        print("get backup data")
        self.getBackupDataUrl = "https://cbr.{}.myhuaweicloud.com/v3/{}/backups/{}".format(self.projectName,
                                                                                           self.projectId,
                                                                                           self.backupId)
        try:
            response = requests.get(url=self.getBackupDataUrl, headers=self.header, verify=False)
            # 检查请求是否成功
            response.raise_for_status()
            # 获取响应数据
            result = json.loads(str(response.content, encoding="utf-8"))
            # 解析响应数据
            return {
                "code": 0,
                "backups": result
            }
        except requests.exceptions.RequestException as e:
            print("get backup data error: ", e)
            return {
                "code": 1,
                "msg": "get backup data error"
            }

class CreateServer:
    """
    创建服务器
    """

    def __init__(self, token, projectName, projectId, backupId, backups, imageRef, flavorRef, vpcId,
                 subnetId, enterpriseProjectId, securityGroups, keyName) -> None:
        self.token = token
        self.header = {
            'Content-Type': 'application/json', 'X-Auth-Token': self.token
        }
        self.projectName = projectName
        self.projectId = projectId
        self.backupId = backupId
        self.backups = backups
        self.imageRef = imageRef
        self.flavorRef = flavorRef
        self.vpcId = vpcId
        self.subnetId = subnetId
        self.enterpriseProjectId = enterpriseProjectId
        self.securityGroups = securityGroups
        self.keyName = keyName

    def transfor_metadata_to_server(self) -> None:
        """
        构造server参数，返回backup_volumes_attached
        """
```

```
        print("transfor metadata to server")
        backup = self.backups["backup"]
        root_volume, data_volumes = handle_backup_volumes(backup["children"])
        # 构建创建服务器的参数
        payload = {
            "server": {
                "name": backup["name"],
                "imageRef": self.imageRef,
                "root_volume": root_volume,
                "data_volumes": data_volumes,
                "flavorRef": self.flavorRef,
                "vpcid": self.vpcId,
                "security_groups": self.securityGroups,
                "nics": [{
                    "subnet_id": self.subnetId
                }
                ],
                "key_name": self.keyName,
                "count": 1,
                "extendparam": {
                    "enterprise_project_id": self.enterpriseProjectId
                }
            }
        }
        self.payload = payload
        print(payload)

    def create_server(self) -> dict:
        print("create server")
        self.createServerUrl = "https://ecs.{}.myhuaweicloud.com/v1/{}/cloudservers".format(self.projectName,
self.projectId)
        json_data = json.dumps(self.payload)
        try:
            response = requests.post(self.createServerUrl,
                                     data=json_data,
                                     headers=self.header,
                                     verify=False)
            print(json.loads(str(response.content, encoding="utf-8")))
            # 检查请求是否成功
            response.raise_for_status()
            # 获取响应数据
            result = json.loads(str(response.content, encoding="utf-8"))
            # {'job_id': 'ff8080828ee22cea018f27bdd23c6477', 'serverIds': ['6cb36d34-b111-48be-9577-
b52dbb74adae']}
            return {
                "job_id": result["job_id"],
                "serverIds": result["serverIds"],
                "code": 0
            }
        except requests.exceptions.RequestException as e:
            print("create server error: ", e)
            return {
                "msg": "create server error",
                "code": 1
            }

    def get_server(self, serverId):
        """
        根据id查询ecs信息
        """
        print("get server, serverId is ", serverId)
        getServerdataUrl = "https://ecs.{}.myhuaweicloud.com/v1/{}/cloudservers/{}".format(self.projectName,
                                                                                           self.projectId,
                                                                                           serverId)

        try:
            response = requests.get(url=getServerdataUrl, headers=self.header, verify=False)
            # 检查请求是否成功
            response.raise_for_status()
```



```
        # 获取响应数据
        result = json.loads(str(response.content, encoding="utf-8"))
        # 解析响应数据
        server_volumes_attached = result["server"]["os-extended-volumes:volumes_attached"]
        # [{'id': '66fc89cd-874c-480a-8e49-bd1d361ff0a0', 'delete_on_termination': 'true', 'device': '/dev/
vda', 'bootIndex': '0'}]
        return server_volumes_attached
    except requests.exceptions.RequestException as e:
        print("get server data error: ", e)

def handle_backup_volumes(backupChildren):
    # 单独处理用于构造root_volume和data_volumes
    root_volume = {}
    data_volumes = []
    # 根据Children顺序赋值root_volume和data_volumes
    for child in backupChildren:
        bootable = child["extend_info"]["bootable"]
        volumeSize = max(40, child["resource_size"])
        if bootable:
            root_volume["volumetype"] = config.volumeType
            root_volume["size"] = volumeSize
        else:
            data_volumes.append({
                "volumetype": config.volumeType,
                "size": volumeSize
            })
    return root_volume, data_volumes

def build_restore_mappings(server_volumes_attached, backup_children):
    print("build restore mappings")
    print(server_volumes_attached)
    print(backup_children)
    # 根据bootIndex构建restore的mappings
    restore_mappings = []
    for server_volume in server_volumes_attached:
        server_volume_id = server_volume["id"]
        server_bootIndex = server_volume["bootIndex"]
        backup_child = backup_children[int(server_bootIndex)]
        restore_mappings.append({
            "backup_id": backup_child["id"],
            "volume_id": server_volume_id
        })
    return restore_mappings

class Job:
    """
    获取Job信息
    """

    def __init__(self, token, projectName, projectId, jobId) -> None:
        self.token = token
        self.header = {
            'Content-Type': 'application/json', 'X-Auth-Token': self.token
        }
        self.projectName = projectName
        self.projectId = projectId
        self.jobId = jobId
        self.getJobUrl = "https://ecs.{}.myhuaweicloud.com/v1/{}/jobs/{}".format(self.projectName,
                                                                                 self.projectId,
                                                                                 self.jobId)

    def get_job_status(self) -> dict:
        """
        根据jobId查询任务信息
        """
        print("get job status, jobId is ", self.jobId)
        try:
            response = requests.get(url=self.getJobUrl, headers=self.header, verify=False)
            # 检查请求是否成功
```



```
            response.raise_for_status()
            # 获取响应数据
            result = json.loads(str(response.content, encoding="utf-8"))
            # 解析响应数据
            return {
                "code": 0,
                "status": result["status"],
            }
        except requests.exceptions.RequestException as e:
            print("get job status error: ", e)
            return {
                "code": 1,
                "msg": "get job status error",
            }

class Restore:
    """
    恢复备份数据
    """

    def __init__(self, token, projectName, projectId, backupId, serverId, restore_mappings) -> dict:
        self.token = token
        self.header = {
            'Content-Type': 'application/json', 'X-Auth-Token': self.token
        }
        self.projectName = projectName
        self.projectId = projectId
        self.backupId = backupId
        self.serverId = serverId
        self.restore_mappings = restore_mappings
        self.backupRestoreUrl = "https://cbr.{}.myhuaweicloud.com/v3/{}/backups/{}/
restore".format(self.projectName,
                                                                        self.projectId,
                                                                        self.backupId)

    def backup_restore(self) -> dict:
        """
        恢复备份数据
        """
        print("backup restore, backupId is {}, serverId is {}".format(self.backupId, self.serverId))
        payload = {
            "restore": {
                "mappings": self.restore_mappings,
                "power_on": True,
                "server_id": self.serverId
            }
        }
        json_data = json.dumps(payload)
        try:
            response = requests.post(self.backupRestoreUrl,
                                     data=json_data,
                                     headers=self.header,
                                     verify=False)
            # 检查请求是否成功
            response.raise_for_status()
            # 获取响应数据
            result = json.loads(str(response.content, encoding="utf-8"))
            # {}
            return {
                "code": 0
            }
        except requests.exceptions.RequestException as e:
            print("backup restore error: ", e)
            return {
                "code": 1,
                "msg": "backup restore error",
            }

def http_error(code):
```

```
    return code == 1

if __name__ == '__main__':
    # IAM鉴权获取token
    print("start to get token")
    tokenObj = IamLogin(config.iamDomain, config.iamUser, config.iamPassword,
config.projectName).get_auth_token()
    if http_error(tokenObj["code"]):
        print("backup failed, msg is {}".format(tokenObj["msg"]))
    else:
        token = tokenObj["token"]
        # 记录创建服务器的job信息
        jobInfos = []
        # 根据备份副本ID列表，批量恢复数据
        for backupId in config.backupIds:
            print("start the backup process, backup id is ", backupId)
            resultData = {"backupId": backupId}
            # 初始化GetBackup参数
            get_backup = GetBackup(token, config.projectName, config.projectId, backupId)
            # 根据backupId查询备份元数据
            backupsObj = get_backup.get_backup()
            if http_error(backupsObj["code"]):
                resultData["status"] = "FAILED"
                resultData["msg"] = backupsObj["msg"]
                print("backup process failed, msg is {}".format(backupsObj["msg"]))
                jobInfos.append(resultData)
                continue
            else:
                backups = backupsObj["backups"]
                # 初始化CreateServer参数
                create_server = CreateServer(token, config.projectName, config.projectId, backupId, backups,
                                             config.imageRef, config.flavorRef, config.vpcId,
                                             config.subnetId, config.enterpriseProjectId, config.securityGroups,
                                             config.keyName)
                create_server.transfor_metadata_to_server()
                # 创建服务器
                createServerData = create_server.create_server()
                if http_error(createServerData["code"]):
                    resultData["status"] = "FAILED"
                    resultData["msg"] = createServerData["msg"]
                    print("backup process failed, msg is {}".format(createServerData["msg"]))
                    jobInfos.append(resultData)
                    continue
                else:
                    serverId = createServerData["serverIds"][0]
                    resultData["serverId"] = serverId
                    resultData["status"] = "START"
                    # 监控创建服务器任务状态
                    print("start to get create server job status, please wait...")
                    get_job = Job(token, config.projectName, config.projectId, createServerData["job_id"])
                    while True:
                        time.sleep(config.delayInSeconds)
                        jobStatusObj = get_job.get_job_status()
                        if http_error(jobStatusObj["code"]):
                            resultData["status"] = "FAILED"
                            resultData["msg"] = createServerData["msg"]
                            print("backup process failed, msg is {}".format(createServerData["msg"]))
                            jobInfos.append(resultData)
                            break
                        else:
                            jobStatus = jobStatusObj["status"]
                            if jobStatus == "SUCCESS":
                                createServerData["status"] = "SUCCESS"
                                print("The create server job is executed successfully.")
                                break
                            elif jobStatus == "RUNNING" or jobStatus == "INIT":
                                print("job status is {}, please wait...".format(jobStatus))
                            if jobStatus is None or jobStatus == "FAIL":
                                print("The job failed.")
```

```
                        break
                # 恢复备份数据
                if createServerData["status"] == "SUCCESS":
                    # 根据bootIndex构建restore的mappings
                    server_volumes_attached = create_server.get_server(serverId)
                    backup_children = backups["backup"]["children"]
                    restore_mappings = build_restore_mappings(server_volumes_attached, backup_children)
                    restore = Restore(token, config.projectName, config.projectId, backupId, serverId,
restore_mappings)
                    restoreObj = restore.backup_restore()
                    if http_error(restoreObj["code"]):
                        resultData["status"] = "FAILED"
                        resultData["msg"] = restoreObj["msg"]
                        print("backup process failed, msg is {}".format(restoreObj["msg"]))
                        jobInfos.append(resultData)
                    else:
                        resultData["status"] = "SUCCESS"
                        jobInfos.append(resultData)
                else:
                    resultData["status"] = "FAILED"
                    resultData["msg"] = createServerData["msg"]
                    print("backup process failed, msg is {}".format(createServerData["msg"]))
                    jobInfos.append(resultData)
                    break
    # 输出成功和失败的数据
    print(jobInfos)
    successJobList = []
    failedJobList = []
    for jobInfo in jobInfos:
        if jobInfo["status"] == "SUCCESS":
            successJobList.append(jobInfo)
        else:
            failedJobList.append(jobInfo)
    print("backup process end")
    print({
        "successJobList": successJobList,
        "failedJobList": failedJobList
    })
```

执行完成后，回显信息如下：

```
backup process end
{'successJobList': [{'backupId': '0753ffd5-5b2f-4acb-a055-9446cb265e43', 'serverId': 'adf8978f-053f-49f2-9d80-68cdb18c7a03',
  'status': 'SUCCESS'}, {'backupId': 'f85e9978-4eef-4707-a73c-8015635c2af2', 'serverId': 'a299a60e-6d15-4a73-8258-007ca9c2795c',
  'status': 'SUCCESS'}, {'backupId': '4534750e-268e-4d64-90aa-b76dd703f484', 'serverId': '678f9afa-58fb-443a-9e93-8655d6f43342',
  'status': 'SUCCESS'}, {'backupId': '2440b565-cff1-47f6-85b1-22199e0882a3', 'serverId': 'aa8436e7-b217-42b4-9386-fd9920578efe',
  'status': 'SUCCESS'}], 'failedJobList': []}
```

说明

其中，successJobList为触发备份数据恢复成功的列表，failedJobList为触发备份数据恢复失败的列表。

备份恢复结果验证操作

- **备份副本列表**

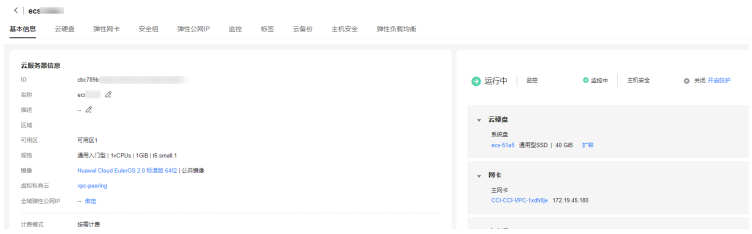
1. 登录管理控制台。
2. 单击管理控制台左上角的，选择区域。
3. 单击“”，选择“存储 > 云备份 CBR”。
4. 左侧导航栏选择“混合云备份 > VMware备份”，备份副本列表的状态下显示“正在恢复”。恢复完成后，状态变为“可用”。

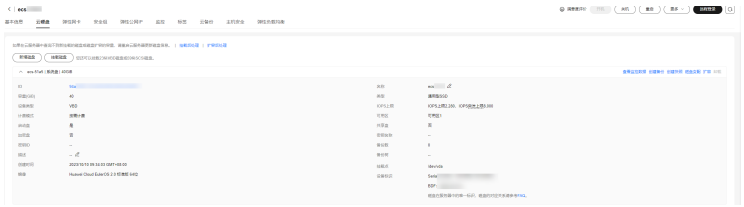
- **云服务器列表**

1. 单击“ ”，选择“计算 > 弹性云服务器ECS”。
2. 左侧导航栏选择“弹性云服务器 > 弹性云服务器”，云服务器列表的状态下显示“运行中”。

- **云服务器磁盘详情**

在弹性云服务器控制台界面，点击对应的云服务器名称，查看云服务器磁盘数据均正常。





执行回滚

前提条件

1. config.py中的回滚参数（serverIds参数）已经配置完毕。
2. 备份数据恢复流程已经执行完毕。

执行回滚操作脚本

通过命令python .\rollback.py执行rollback.py文件，开始备份数据恢复主流程。

rollback.py示例如下：

```
import requests
import json
import time
```



```
import config


class IamLogin:
    """
    login by iam, return token
    """

    def __init__(self, iamDomain: str, iamUser: str, iamPassword: str, projectName: str):
        self.iamDomain = iamDomain
        self.iamUser = iamUser
        self.iamPassword = iamPassword
        self.projectName = projectName
        self.iamTokenUrl = "https://iam.{}.myhuaweicloud.com/v3/auth/tokens".format(self.projectName)

    def get_auth_token(self) -> dict:
        """
        return token
        """
        payload = {
            "auth": {
                "identity": {
                    "methods": [
                        "password"
                    ],
                    "password": {
                        "user": {
                            "domain": {
                                "name": self.iamDomain
                            },
                            "name": self.iamUser,
                            "password": self.iamPassword
                        }
                    }
                },
                "scope": {
                    "project": {
                        "name": self.projectName
                    }
                }
            }
        }
        json_data = json.dumps(payload)
        try:
            response = requests.post(self.iamTokenUrl,
                                     data=json_data,
                                     headers={'Content-Type': 'application/json'},
                                     verify=False)
            # 检查请求是否成功
            response.raise_for_status()
            # 获取响应头的token
            access_token = response.headers.get("X-Subject-Token")
            return {
                "token": access_token,
                "code": 0
            }
        except requests.exceptions.RequestException as e:
            print("get token error: ", e)
            return {
                "msg": "get token error",
                "code": 1
            }

class Rollback:
    """
    服务器回滚
    """

    def __init__(self, token, projectName, projectId, serverIds, deletePublicIp, deleteVolume) -> None:
```

```
        self.token = token
        self.projectName = projectName
        self.projectId = projectId
        self.header = {
            'Content-Type': 'application/json',
            'X-Project-Id': self.projectId,
            'X-Auth-Token': self.token
        }
        self.serverIds = serverIds
        self.deletePublicIp = deletePublicIp
        self.deleteVolume = deleteVolume
        self.deleteServerUrl = "https://ecs.{}.myhuaweicloud.com/v1/{}/cloudservers/
delete".format(self.projectName,
                                                                    self.projectId)

    def delete_cloudservers(self) -> dict:
        """
        根据指定的云服务器ID列表，删除云服务器。可以单个删除，也可以批量删除。
        """
        print("batch delete servers, serverIds are ", self.serverIds)
        if len(self.serverIds) == 0:
            return {
                "code": 1,
                "msg": "serverIds is empty"
            }
        servers = []
        for serverId in self.serverIds:
            servers.append({"id": serverId})
        payload = {
            "servers": servers,
            "delete_publicip": self.deletePublicIp,
            "delete_volume": self.deleteVolume
        }
        json_data = json.dumps(payload)
        try:
            response = requests.post(self.deleteServerUrl,
                                     data=json_data,
                                     headers=self.header,
                                     verify=False)
            # 检查请求是否成功
            response.raise_for_status()
            # 获取响应数据
            result = json.loads(str(response.content, encoding="utf-8"))
            # {'job_id': 'ff8080828ee21983018f27b42f310e0f'}
            # 解析响应数据
            return {
                "code": 0,
                "job_id": result["job_id"]
            }
        except requests.exceptions.RequestException as e:
            print("batch delete servers error: ", e)
            return {
                "code": 1,
                "msg": "batch delete servers error"
            }

class Job:
    """
    获取Job信息
    """

    def __init__(self, token, projectName, projectId, jobId) -> None:
        self.token = token
        self.header = {
            'Content-Type': 'application/json', 'X-Auth-Token': self.token
        }
        self.projectName = projectName
        self.projectId = projectId
        self.jobId = jobId
```

```
        self.getJobUrl = "https://ecs.{}.myhuaweicloud.com/v1/{}/jobs/{}".format(self.projectName,
                                                                                 self.projectId,
                                                                                 self.jobId)

    def get_job_status(self) -> dict:
        """
        根据jobId查询任务信息
        """
        print("get job status, jobId is ", self.jobId)
        try:
            response = requests.get(url=self.getJobUrl, headers=self.header, verify=False)
            # 检查请求是否成功
            response.raise_for_status()
            # 获取响应数据
            result = json.loads(str(response.content, encoding="utf-8"))
            # 解析响应数据
            return {
                "code": 0,
                "status": result["status"],
            }
        except requests.exceptions.RequestException as e:
            print("get job status error: ", e)
            return {
                "code": 1,
                "msg": "get job status error",
            }

def http_error(code):
    return code == 1

if __name__ == '__main__':
    # IAM鉴权获取token
    print("start to get token")
    tokenObj = IamLogin(config.iamDomain, config.iamUser, config.iamPassword,
config.projectName).get_auth_token()
    if http_error(tokenObj["code"]):
        print("rollback failed, msg is {}".format(tokenObj["msg"]))
    else:
        token = tokenObj["token"]
        # 根据服务器ID列表，批量删除数据
        print("start to rollback")
        rollback = Rollback(token, config.projectName, config.projectId, config.serverIds, config.deletePublicip,
                            config.deleteVolume)
        jobObj = rollback.delete_cloudservers()
        if http_error(jobObj["code"]):
            print("rollback failed, msg is {}".format(jobObj["msg"]))
        else:
            get_job = Job(token, config.projectName, config.projectId, jobObj["job_id"])
            # 监控任务状态
            print("start to get rollback job status, please wait...")
            while True:
                time.sleep(config.delayInSeconds)
                jobStatusObj = get_job.get_job_status()
                if http_error(jobStatusObj["code"]):
                    print("rollback failed, msg is {}".format(jobStatusObj["msg"]))
                    break
                else:
                    jobStatus = jobStatusObj["status"]
                    if jobStatus == "SUCCESS":
                        print("The job is executed successfully.")
                        break
                    elif jobStatus == "RUNNING" or jobStatus == "INIT":
                        print("job status is {}, please wait...".format(jobStatus))
                    if jobStatus is None or jobStatus == "FAIL":
                        print("The job failed.")
                        break
```

执行完成后，回显信息如下


```
The job is executed successfully.
```

回滚结果验证

在弹性云服务器控制台界面，左侧导航栏选择“弹性云服务器 > 弹性云服务器”，云服务器已被清退。