Cloud Backup and Recovery

Best Practices

Issue 01

Date 2020-11-06

HUAWEI TECHNOLOGIES CO., LTD.

Contents

1 Using a Custom Script to Implement Application-Consistent Backup

1.1 Using a Custom Script to Implement Consistent Backup for MySQL

1.1.1 Preparations

The following example uses single-server MySQL 5.5 running on SUSE 11 SP3 to demonstrate how to use a custom script to freeze and unfreeze the MySQL database in order to implement application-consistent backup.

Context

An enterprise purchases Elastic Cloud Servers (ECSs) and installs MySQL 5.5 on ECSs for storing business data. As data increases, crash-consistent backup cannot meet the recovery time objective (RTO) and recovery point time (RPO) requirements and therefore application-consistent backup is needed.

Required Data

Table 1-1 Required data

Item	Description	Example
MySQL username	Username for connecting to the MySQL database	root
MySQL password	Password for connecting to the MySQL database	Example@123

1.1.2 Procedure

Step 1 Encrypt the MySQL password.

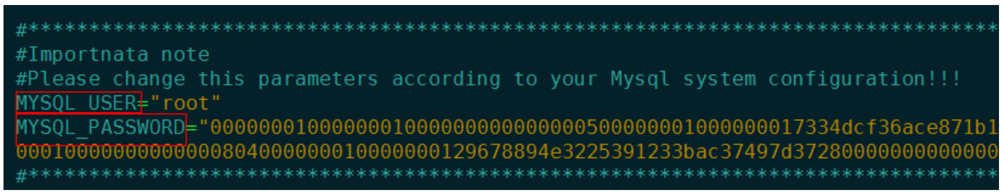
1. Log in to the MySQL server and run the **cd /home/rdadmin/Agent/bin/** command to go to the Agent directory.
2. Run the **/home/rdadmin/Agent/bin/agentcli encpwd** command. The following information is displayed:

```
Enter password:
```

   Enter the MySQL password and press **Enter**. After the encrypted password is displayed, copy it to the clipboard.

Step 2 Run the **cd /home/rdadmin/Agent/bin/thirdparty/ebk_user** command to go to the directory saving the custom scripts and run the **vi mysql_freeze.sh** command to open the example MySQL freezing script.

The following figure shows an example. You need to set **MYSQL_USER** and **MYSQL_PASSWORD** based on your actual conditions, where **MYSQL_PASSWORD** should be the encrypted password obtained in **step 1**.



```
#******************************************************************************
#Importnata note
#Please change this parameters according to your Mysql system configuration!!!
MYSQL_USER="root"
MYSQL_PASSWORD="00000001000000010000000000000000500000001000000017334dcf36ace871b1
0001000000000000080400000001000000129678894e3225391233bac37497d3728000000000000000
#******************************************************************************
```

You can also run the **sed** command to modify the configuration:

sed -i 's/^MYSQL_PASSWORD=.*/MYSQL_PASSWORD="*XXX*"/' mysql_freeze.sh mysql_unfreeze.sh, where ***XXX*** indicates the password obtained in step 1.

If you run this command, both the freezing and unfreezing scripts will be modified and therefore step 3 is not needed.

Step 3 Run the **vi mysql_unfreeze.sh** command to open the example MySQL unfreezing script and change the username and password in the script to be consistent with your actual settings.

The **mysql_unfreeze.sh** and **mysql_freeze.sh** scripts can only be used to freezing and unfreezing databases. If other operations are required, you can add them in the scripts via compilation. For details, see **1.3 Using a Custom Script to Implement Consistent Backup for Other Linux Applications**.

⚠ CAUTION

MySQL is frozen by running the **FLUSH TABLES WITH READ LOCK** command. This command will not trigger disk flushing on **bin log**. If **bin log** is enabled and the value of **sync_binlog** is not **1**, some SQL operations saved in the backup image may not be recorded in **bin log**. To realize complete protection on **bin log**, set **sync_binlog** to **1**.

----End

1.2 Using a Custom Script to Implement Consistent Backup for SAP HANA

1.2.1 Preparations

The following example uses single-server HANA 2.0 running on SUSE 11 SP4 for SAP to demonstrate how to use a custom script to freeze and unfreeze the HANA database in order to implement database backup.

Context

An enterprise purchases ECSs and installs HANA 2.0 on ECSs for saving business data. As data increases, crash-consistent backup cannot meet the RTO and RPO requirements and therefore application-consistent backup is needed.

Required Data

Table 1-2 Required data

Item	Description	Example
HANA username	Username for connecting to the HANA SYSTEMDB database	system
HANA password	Password for connecting to the HANA SYSTEMDB database	Example@123
HANA instance ID	Instance ID for connecting to the HANA database	00
HANA SID	SID for connecting to the HANA database	WXJ

1.2.2 Procedure

Step 1 Encrypt the HANA password.

1. Log in to the HANA server and run the **cd /home/rdadmin/Agent/bin/** command to go to the Agent directory.

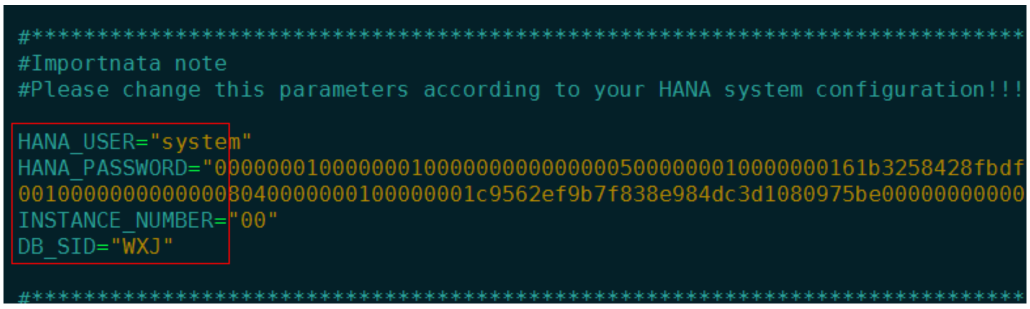
2. Run the **/home/rdadmin/Agent/bin/agentcli encpwd** command. The following information is displayed:

```
Enter password:
```

   Enter the HANA password and press **Enter**. After the encrypted password is displayed, copy it to the clipboard.

   Run the **cd /home/rdadmin/Agent/bin/thirdparty/ebk_user** command to go to the custom script directory and run the **vi hana_freeze.sh** command to open the example freezing script.

Step 2 The following figure shows an example. You need to set **HANA_USER**, **HANA_PASSWORD**, and **INSTANCE_NUMBER DB_SID** based on your actual conditions, where **HANA_PASSWORD** should be the encrypted password obtained in step 1.



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

HANA_USER="system"
HANA_PASSWORD="00000001000000010000000000000000500000001000000161b3258428fbdf
0010000000000000804000000001000000001c9562ef9b7f838e984dc3d1080975be00000000000
INSTANCE_NUMBER="00"
DB_SID="WXJ"

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

You can also run the **sed** commands to modify the configuration:

sed -i 's/^HANA_USER=.*/HANA_USER="*XXX*"/' hana_freeze.sh hana_unfreeze.sh, where ***XXX*** indicates the database username.

sed -i 's/^HANA_PASSWORD=.*/HANA_PASSWORD="*XXX*"/' hana_freeze.sh hana_unfreeze.sh, where ***XXX*** indicates the password obtained in step 1

sed -i 's/^INSTANCE_NUMBER=.*/INSTANCE_NUMBER="*XXX*"/' hana_freeze.sh hana_unfreeze.sh, where ***XXX*** indicates the database username

sed -i 's/^DB_SID=.*/DB_SID="*XXX*"/' hana_freeze.sh hana_unfreeze.sh, where ***XXX*** indicates the database SID

If you run this command, both the freezing and unfreezing scripts will be modified and therefore **step 3** is not needed.

Step 3 Run the **vi hana_unfreeze.sh** command to open the example HANA unfreezing script and change the username, password, instance ID, and SID in the script to be consistent with your actual settings.

The **hana_freeze.sh** and **hana_unfreeze.sh** scripts can only be used to freezing and unfreezing databases. If other operations are required, you can add them in the scripts via compilation. For details, see **1.3 Using a Custom Script to Implement Consistent Backup for Other Linux Applications**.

WARNING

When freezing the SAP HANA database, you need to freeze the XFS file systems of the data volumes as SAP suggested. Otherwise, data inconsistency may occur. The example script mentioned in this section will query the mount point of the **Data** volume used by the HANA database and then use the **xfs_freeze** command to freeze the database.

If the HANA system does not have an independent partition for saving the data volumes as SAP suggested but stores them in the same partition as the system volume, modify the **hana_freeze.sh** script by commenting out lines related to **xfs_freeze** to avoid the freezing of the entire system. However, such operations still could not eliminate data inconsistency.

----End

1.3 Using a Custom Script to Implement Consistent Backup for Other Linux Applications

1.3.1 Context

If other Linux applications need application-consistent backup, you can compile custom scripts to freeze and unfreeze them. To ensure the custom scripts invokable by the Agent, save them in the **/home/rdadmin/Agent/bin/thirdparty/ebk_user** directory.

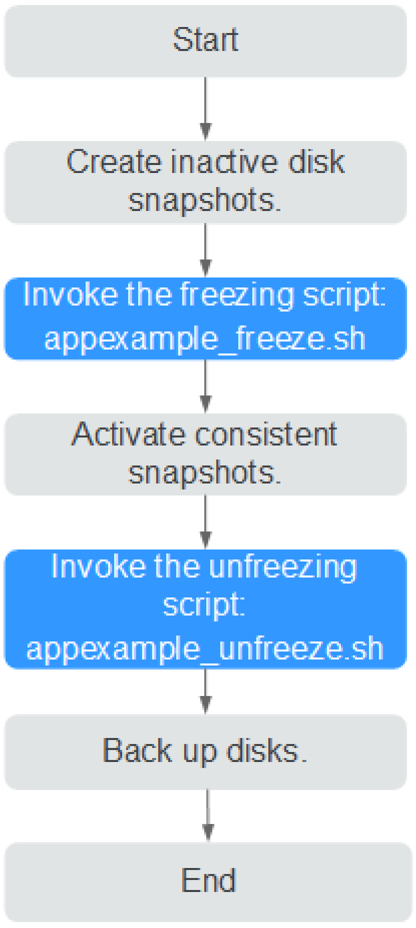
The following example uses an application named **appexample** for demonstration.

appexample is a new database. It provides the **appexample -freeze** and **appexample -unfreeze** commands for freezing and unfreezing.

To implement application-consistent backup, you need to compile two scripts named **appexample_freeze.sh** and **appexample_unfreeze.sh**. During a backup, the Agent first invokes the **appexample_freeze.sh** script to freeze I/Os, then activates consistent snapshots on disks to ensure that the backup data is consistent, and finally invokes **appexample_unfreeze.sh** to unfreeze I/Os.

Figure 1-1 shows the backup process.

Figure 1-1 Application-consistent backup flowchart


Start
Create inactive disk snapshots.
Invoke the freezing script: appexample_freeze.sh
Activate consistent snapshots.
Invoke the unfreezing script: appexample_unfreeze.sh
Back up disks.
End

1.3.2 Compiling a Freezing Script

Example freezing script named **appexample_freeze.sh**:

```
#!/bin/sh
AGENT_ROOT_PATH=$1  #The root directory required when the Agent invokes the script. Functions, such as
log functions, will use this variable. Do not rename this directory.
PID=$2 #The PID required when the Agent invokes the script. It is used for command output and do not
rename it.
. "${AGENT_ROOT_PATH}/bin/agent_func.sh"#Reference script framework, which provides functions, such
as logging, encryption, and decryption
#Result processing function, which writes operation results into given files for invokers to obtain return
values.
 # Input parameter. $1: 0 indicates a success; 1 indicates a failure.
# No return value
#RESULT_FILE is defined in agent_func.sh.
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."
    #Check whether appexample exists. If not, 0 is returned and the script exits.
    #In the process of freezing I/Os, the Agent program invokes each freezing script in sequence. If any script
fails to be invoked, the whole process fails. To avoid interference from other programs, 0 should be returned
when appexample cannot be found.
    which appexample
    if [ $? -ne 0 ]
    then
            Log "[INFO]:appexample is not installed."
            ExitWithResult 0
    fi
    #Invoke the actual freezing command.
    appexample -freeze
    if [ $? -ne 0 ]
    then
            Log "[INFO]:appexample freeze failed."
            #Freezing failed. Record the result and exit.
            ExitWithResult 1
    fi
    Log "[INFO]:Freeze appexample success."
    #Freezing successful. Record the result and exit.
    ExitWithResult 0
}
Main
```

1.3.3 Compiling an Unfreezing Script

Example unfreezing script named **appexample_unfreeze.sh**:

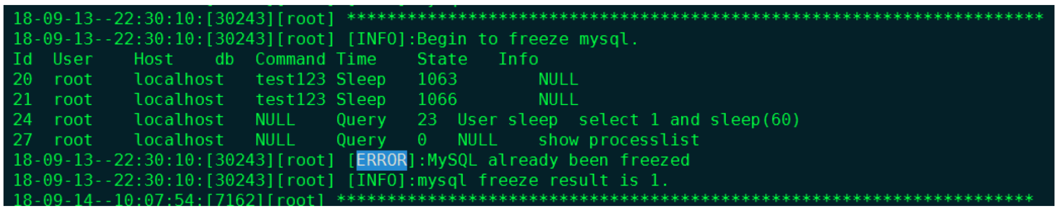
```
#!/bin/sh
AGENT_ROOT_PATH=$1  #The root directory required when the Agent invokes the script. Functions, such as
log functions, will use this variable. Do not rename this directory.
PID=$2 #The PID required when the Agent invokes the script. It is used for command output and do not
rename it.
. "${AGENT_ROOT_PATH}/bin/agent_func.sh"#Reference script framework, which provides functions, such
as logging, encryption, and decryption
#Result processing function, which writes operation results into given files for invokers to obtain return
values.
 # Input parameter. $1: 0 indicates a success; 1 indicates a failure.
# No return value
#RESULT_FILE is defined in agent_func.sh.
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."
    #Check whether appexample exists. If not, 0 is returned and the script exits.
    #In the process of unfreezing I/Os, the Agent program invokes each unfreezing script in sequence. If any
script fails to be invoked, the whole process fails. To avoid interference from other programs, 0 should be
returned when appexample cannot be found.
    which appexample
    if [ $? -ne 0 ]
    then
            Log "[INFO]:appexample is not installed."
            ExitWithResult 0
    fi
    #Invoke the actual unfreezing command.
    appexample -unfreeze
    if [ $? -ne 0 ]
    then
         Log "[INFO]:appexample freeze failed."
        #Unfreezing failed. Record the result and exit.
         ExitWithResult 1
    fi
    Log "[INFO]:Freeze appexample. success"
    #Unfreezing successful. Record the result and exit.
    ExitWithResult 0
}
Main
```

1.4 Troubleshooting a Custom Script Error

Application-consistent backup may fail due to custom script defects. In such conditions, open the **/home/rdadmin/Agent/log/thirdparty.log** file and view logs to locate the fault.

Figure 1-2 provides a log example recording a MySQL database freezing failure

Figure 1-2 Log example



```
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] **********************************************************************
```

18-09-13--22:30:10 in the first column records the logging time.

[30243] in the second column is the script PID.

[root] in the third column is the user who executes the script.

[INFO] or **[ERROR]** in the fourth column indicates the log level.

When a script invocation failure occurs, you can view the **ERROR** logs generated around the failure occurrence time to locate the fault. In **Figure 1-2**, the freezing fails because the MySQL database is in the frozen state and cannot be frozen again.

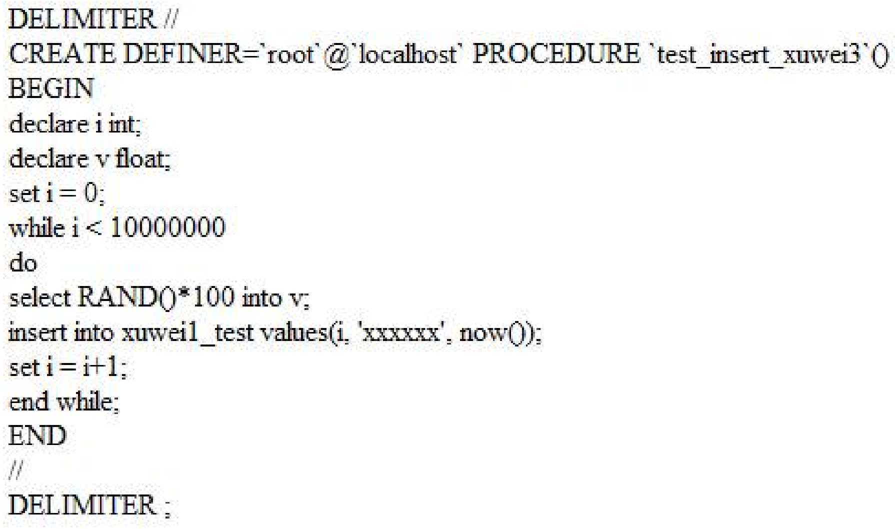
1.5 Verifying the Application-Consistent Backup Result (Linux)

After application-consistent backup is implemented using customized scripts, perform the following operations to check whether the backup is successful: This section uses the MySQL database as an example.

Step 1 Log in to MySQL database and create a database.

Step 2 After the database is created, create a stored procedure. For details, see **Figure 1-3**.

Figure 1-3 Creating a stored procedure



```
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 ;
```

Step 3 Log in to CBR Console and create an application-consistent backup for the desired ECS.

Step 4 After the backup is complete, open the **/home/rdadmin/Agent/log/rdagent.log** file and view the freezing and unfreezing logs to determine the freezing and unfreezing times.

Step 5 Use the newly created application-consistent backup to restore the ECS. After the restoration is successful, log in to the ECS and database and check the time when the last data record is inserted.

Step 6 Compare the VSS freezing success time recorded in **step 5** with the time recorded in **step 4**. Before the freezing is successful, data insertion is stopped. Therefore, the time in **step 5** should be earlier than that in **step 4**. If the time in **step 5** is earlier than that in **step 4**, the backup is successful.

----End

1.6 Verifying the Application-Consistent Backup Result (Windows)

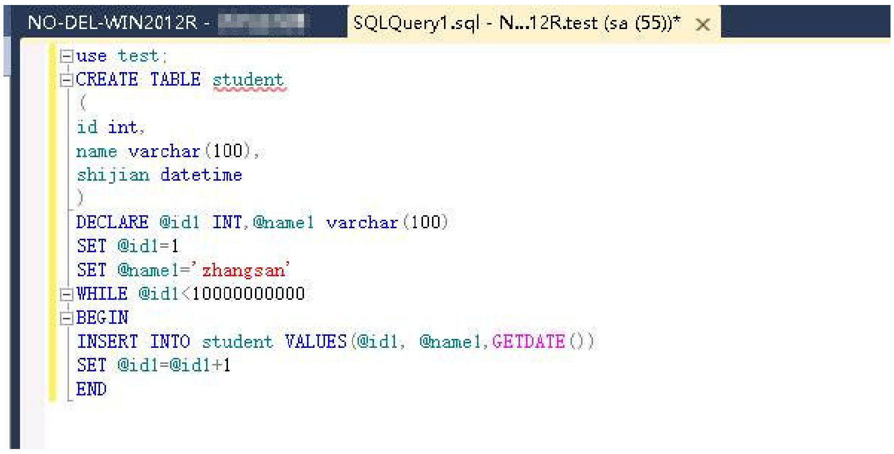
After application-consistent backup is implemented using customized scripts, perform the following operations to check whether the backup is successful: This section uses the SQL_SERVER database as an example.

Procedure

Step 1 Log in to SQL_SERVER database and create a database.

Step 2 After the database is created, create a stored procedure. For details, see **Figure 1-4**.

Figure 1-4 Creating a stored procedure



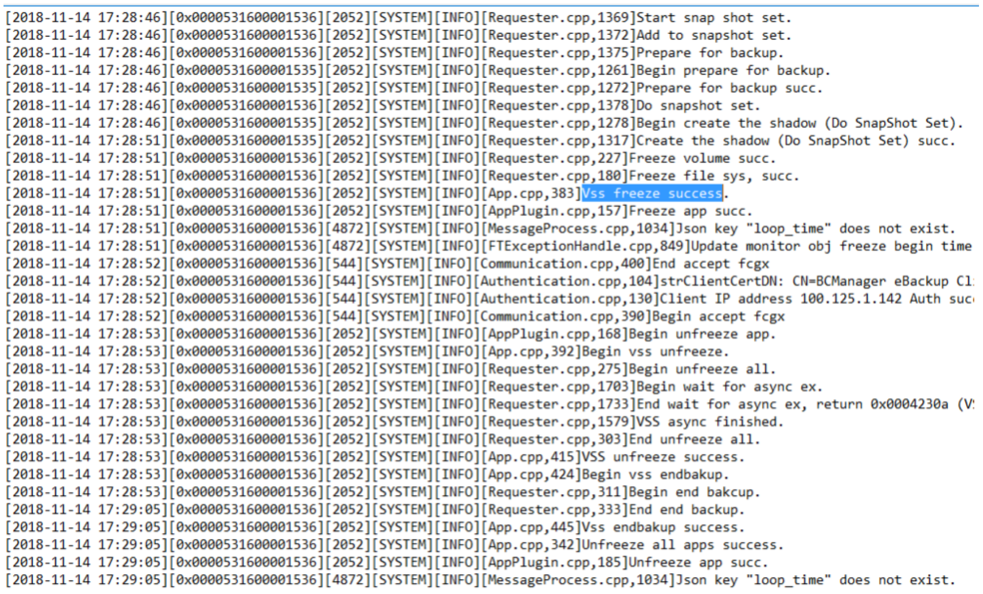
NO-DEL-WIN2012R - [redacted]   SQLQuery1.sql - N...12R.test (sa (55))*

```
use test;
CREATE TABLE student
(
id int,
name varchar(100),
shijian datetime
)
DECLARE @id1 INT,@name1 varchar(100)
SET @id1=1
SET @name1='zhangsan'
WHILE @id1<10000000000
BEGIN
INSERT INTO student VALUES(@id1, @name1,GETDATE())
SET @id1=@id1+1
END
```

Step 3 Log in to CBR Console and create an application-consistent backup for the desired ECS.

Step 4 After the backup is complete, open the **Cloud Server Backup Agent-WIN64\log\rdagent.txt** file and view the freezing and unfreezing logs to determine the freezing and unfreezing times. As shown in the figure, the freeze success time is **17:28:51**.

Figure 1-5 Viewing logs



```
[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.
```

Step 5 Use the newly created application-consistent backup to restore the ECS. After the restoration is successful, log in to the ECS and database and check the time when the last data record is inserted (**17:28:49** in the following figure).

Step 6 Compare the VSS freezing success time recorded in **step 5** with the time recorded in **step 4**. Before the freezing is successful, data insertion is stopped. Therefore, the time in **step 5** should be earlier than that in **step 4**. If the time in **step 5** is earlier than that in **step 4**, the application-consistent backup is successful.

----End

1.7 Protecting SQL Server in Failover Cluster Mode

Currently, cloud server backup provides application-consistent backup only on single VMs. The support for clustered databases will be implemented later.

In Failover Cluster mode, the SQL Server service is enabled only on the active node. Because of this, you only need to associate the active node with the policy when creating a cloud server backup. After an active/standby switchover, you need to modify the policy at once to keep the node being backed up is always the active one. To restore data of the active node, stop all standby nodes first.

1.8 Protecting SQL Server in Always on Availability Groups Mode

Currently, cloud server backup provides application-consistent backup only on single VMs. The support for clustered databases will be implemented later.

In Always on Availability Groups mode, the SQL Server service is enabled both on the active and standby nodes, data is replicated from the active node to standby nodes, and the active node contains the complete data. When creating a cloud server backup, you only need to associate the active node with the policy. After an active/standby switchover, you need to modify the policy at once to keep the node being backed up is always the active one.

Restoring data of the active node triggers synchronization, because of the SQL Server mechanism. The synchronization will overwrite data on the standby nodes, resulting in loss of data generated during the restoration. To prevent such unexpected data loss, we recommend you to perform entire-ECS restoration only when none of the active and standby nodes is available.

A Change History

Released On	Description
2020-11-06	This issue is the first official release.